

PICTURING QUANTUM PROCESSES

A First Course in Quantum Theory and
Diagrammatic Reasoning

BOB COECKE AND ALEKS KISSINGER



PICTURING QUANTUM PROCESSES

A First Course in Quantum Theory and
Diagrammatic Reasoning

BOB COECKE

University of Oxford

ALEKS KISSINGER

Radboud University



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom
One Liberty Plaza, 20th Floor, New York, NY 10006, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
4843/24, 2nd Floor, Ansari Road, Daryaganj, Delhi – 110002, India
79 Anson Road, #06–04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9781107104228
10.1017/9781316219317

© Bob Coecke and Aleks Kissinger 2017

This publication is in copyright. Subject to statutory exception and to the provisions of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published 2017

Printed in the United Kingdom by TJ International Ltd. Padstow Cornwall

A catalogue record for this publication is available from the British Library

Library of Congress Cataloging-in-Publication Data

Names: Coecke, Bob, author. | Kissinger, Aleks, author.

Title: Picturing quantum processes : a first course in quantum theory and diagrammatic reasoning / Bob Coecke (University of Oxford), Aleks Kissinger (Radboud University).

Description: Cambridge, United Kingdom ; New York, NY : Cambridge University Press, 2017. |

Includes bibliographical references and index.

Identifiers: LCCN 2016035537 | ISBN 9781107104228 (hardback ; alk. paper) | ISBN 110710422X (hardback ; alk. paper)

Subjects: LCSH: Quantum theory. | Quantum computing. | Logic, Symbolic and mathematical.

Classification: LCC QC174.12 .C57 2017 | DDC 530.12–dc23 LC record available at <https://lcn.loc.gov/2016035537>

ISBN 978-1-107-10422-8 Hardback

Cambridge University Press has no responsibility for the persistence or accuracy of URLs for external or third-party Internet Web sites referred to in this publication, and does not guarantee that any content on such Web sites is, or will remain, accurate or appropriate.

Contents

<i>Preface</i>	<i>page xiii</i>
1 <u>Introduction</u>	1
1.1 <u>The Penguins and the Polar Bear</u>	1
1.2 <u>So What's New?</u>	5
1.2.1 <u>A New Attitude to Quantum Theory: 'Features'</u>	6
1.2.2 <u>A New Form of Mathematics: 'Diagrams'</u>	9
1.2.3 <u>A New Foundation for Physics: 'Process Theories'</u>	11
1.2.4 <u>A New Paradigm: 'Quantum Picturalism'</u>	13
1.3 <u>Historical Notes and References</u>	15
2 <u>Guide to Reading This Textbook</u>	19
2.1 <u>Who Are You and What Do You Want?</u>	19
2.2 <u>The Menu</u>	20
2.2.1 <u>How Diagrams Evolve in This Book</u>	20
2.2.2 <u>Hollywood-Style Trailer</u>	22
2.2.3 <u>Some Intermediate Symbolic Pollution</u>	24
2.2.4 <u>Summaries, Historical Notes, References, Epigraphs</u>	25
2.2.5 <u>Starred Headings and Advanced Material Sections</u>	25
2.3 <u>FAQ</u>	25
3 <u>Processes as Diagrams</u>	28
3.1 <u>From Processes to Diagrams</u>	29
3.1.1 <u>Processes as Boxes and Systems as Wires</u>	29
3.1.2 <u>Process Theories</u>	32
3.1.3 <u>Diagrams Are Mathematics</u>	35
3.1.4 <u>Process Equations</u>	38
3.1.5 <u>Diagram Substitution</u>	41
3.2 <u>Circuit Diagrams</u>	44
3.2.1 <u>Parallel Composition</u>	44
3.2.2 <u>Sequential Composition</u>	45
3.2.3 <u>Two Equivalent Definitions of Circuits</u>	46

	3.2.4 Diagrams Beat Algebra	50
3.3	Functions and Relations as Processes	52
	3.3.1 Sets	53
	3.3.2 Functions	54
	3.3.3 Relations	56
	3.3.4 Functions versus Relations	59
3.4	Special Processes	59
	3.4.1 States, Effects, and Numbers	59
	3.4.2 Saying the Impossible: Zero Diagrams	66
	3.4.3 Processes That Are Equal ‘Up to a Number’	68
	3.4.4 Dirac Notation	69
3.5	Summary: What to Remember	72
3.6	Advanced Material*	74
	3.6.1 Abstract Tensor Systems*	75
	3.6.2 Symmetric Monoidal Categories*	77
	3.6.3 General Diagrams versus Circuits*	80
3.7	Historical Notes and References	81
4	String Diagrams	83
4.1	Cups, Caps, and String Diagrams	84
	4.1.1 Separability	85
	4.1.2 Process–State Duality	88
	4.1.3 The Yanking Equations	90
	4.1.4 String Diagrams	92
4.2	Transposition and Trace	94
	4.2.1 The Transpose	95
	4.2.2 Transposition of Composite Systems	99
	4.2.3 The Trace and Partial Trace	101
4.3	Reflecting Diagrams	103
	4.3.1 Adjoints	103
	4.3.2 Conjugates	108
	4.3.3 The Inner Product	113
	4.3.4 Unitarity	117
	4.3.5 Positivity	118
	4.3.6 \otimes-Positivity	120
	4.3.7 Projectors	122
4.4	Quantum Features from String Diagrams	125
	4.4.1 A No-Go Theorem for Universal Separability	125
	4.4.2 Two No-Go Theorems for Cloning	129
	4.4.3 As If time Flows Backwards	134

4.4.4	Teleportation	137
4.5	Summary: What to Remember	141
4.6	Advanced Material*	145
4.6.1	String Diagrams in Abstract Tensor Systems*	146
4.6.2	Dual Types and Self-Duality*	146
4.6.3	Dagger Compact Closed Categories*	150
4.7	Historical Notes and References	152
5	Hilbert Space from Diagrams	154
5.1	Bases and Matrices	156
5.1.1	Basis for a Type	156
5.1.2	Matrix of a Process	162
5.1.3	Sums of Processes	167
5.1.4	Processes from Matrices	172
5.1.5	Matrices of Isometries and Unitaries	177
5.1.6	Matrices of Self-Adjoint and Positive Processes	182
5.1.7	Traces of Matrices	185
5.2	Matrix Calculus	187
5.2.1	Sequential Composition of Matrices	187
5.2.2	Parallel Composition of Matrices	188
5.2.3	Matrix Form of Cups and Caps	194
5.2.4	String Diagrams of Matrices	197
5.2.5	Matrices as Process Theories	198
5.3	Hilbert Spaces	200
5.3.1	Linear Maps and Hilbert Spaces from Diagrams	200
5.3.2	Positivity from Conjugation	203
5.3.3	Why Mathematicians Love Complex Numbers	204
5.3.4	Classical Logic Gates as Linear Maps	210
5.3.5	The X-Basis and the Hadamard Linear Map	213
5.3.6	Bell Basis and Bell Maps	218
5.4	Hilbert Spaces versus Diagrams	222
5.4.1	String Diagrams Are Complete for Linear Maps	223
5.4.2	The Set-Theoretic Definition of Hilbert Spaces	226
5.5	Summary: What to Remember	233
5.6	Advanced Material*	238
5.6.1	Beyond Finite Dimensions*	238
5.6.2	Categories with Sums and Bases*	240
5.6.3	Sums in Knot Theory*	242
5.6.4	Equivalence of Symmetric Monoidal Categories*	243
5.7	Historical Notes and References	249

6	<u>Quantum Processes</u>	251
6.1	<u>Pure Quantum Maps from Doubling</u>	253
6.1.1	<u>Doubling Generates Probabilities</u>	253
6.1.2	<u>Doubling Eliminates Global Phases</u>	257
6.1.3	<u>The Process Theory of Pure Quantum Maps</u>	260
6.1.4	<u>Things Preserved by Doubling</u>	265
6.1.5	<u>Things Not Preserved by Doubling</u>	270
6.2	<u>Quantum Maps from Discarding</u>	274
6.2.1	<u>Discarding</u>	275
6.2.2	<u>Impurity</u>	279
6.2.3	<u>Weight and Causality for Quantum States</u>	282
6.2.4	<u>The Process Theory of Quantum Maps</u>	287
6.2.5	<u>Causality for Quantum Maps</u>	292
6.2.6	<u>Isometry and Unitarity from Causality</u>	294
6.2.7	<u>Kraus Decomposition and Mixing</u>	298
6.2.8	<u>The No-Broadcasting Theorem</u>	305
6.3	<u>Relativity in Process Theories</u>	309
6.3.1	<u>Causal Structure</u>	309
6.3.2	<u>Causality Implies Non-signalling</u>	314
6.3.3	<u>Causality and Covariance</u>	315
6.4	<u>Quantum Processes</u>	317
6.4.1	<u>Non-deterministic Quantum Processes</u>	318
6.4.2	<u>Non-deterministic Realisation of All Quantum Maps</u>	322
6.4.3	<u>Purification of Quantum Processes</u>	324
6.4.4	<u>Teleportation Needs Classical Communication</u>	327
6.4.5	<u>Controlled Processes</u>	329
6.4.6	<u>Quantum Teleportation in Detail</u>	331
6.5	<u>Summary: What to Remember</u>	334
6.6	<u>Advanced Material*</u>	337
6.6.1	<u>Doubling General Process Theories*</u>	338
6.6.2	<u>Axiomatizing Doubling*</u>	339
6.6.3	<u>And Now for Something Completely Different*</u>	342
6.7	<u>Historical Notes and References</u>	343
7	<u>Quantum Measurement</u>	345
7.1	<u>ONB Measurements</u>	347
7.1.1	<u>A Dodo's Introduction to Measurement Devices</u>	347
7.1.2	<u>Demolition ONB Measurements</u>	350
7.1.3	<u>Non-demolition ONB Measurements</u>	355
7.1.4	<u>Superposition and Interference</u>	357
7.1.5	<u>The Next Best Thing to Observation</u>	360

7.2	Measurement Dynamics and Quantum Protocols	361
7.2.1	Measurement-Induced Dynamics I: Backaction	362
7.2.2	Example: Gate Teleportation	365
7.2.3	Measurement-Induced Dynamics II: Collapse	366
7.2.4	Example: Entanglement Swapping	369
7.3	More General Species of Measurement	371
7.3.1	Von Neumann Measurements	371
7.3.2	Von Neumann's Quantum Formalism	377
7.3.3	POVM Measurements	380
7.3.4	Naimark and Ozawa Dilation	383
7.4	Tomography	385
7.4.1	State Tomography	385
7.4.2	Informationally Complete Measurements	388
7.4.3	Local Tomography = Process Tomography	390
7.5	Summary: What to Remember	392
7.6	Advanced Material*	396
7.6.1	Do Quantum Measurements Even Exist?*	396
7.6.2	Projectors and Quantum Logic*	399
7.6.3	Failure of Local Tomography*	401
7.7	Historical Notes and References	402
8	Picturing Classical-Quantum Processes	405
8.1	Classical Systems as Wires	409
8.1.1	Double versus Single Wires	410
8.1.2	Example: Dense Coding	413
8.1.3	Measurement and Encoding	415
8.1.4	Classical-Quantum Maps	416
8.1.5	Deleting and Causality	421
8.2	Classical Maps from Spiders	423
8.2.1	Classical Maps	424
8.2.2	Copying and Deleting	427
8.2.3	Spiders	437
8.2.4	If It behaves like a Spider It Is One	444
8.2.5	All Linear Maps as Spiders + Isometries	446
8.2.6	Spider Diagrams and Completeness	451
8.3	Quantum Maps from Spiders	453
8.3.1	Measuring and Encoding as Spiders	454
8.3.2	Decoherence	459
8.3.3	Classical, Quantum, and Bastard Spiders	463
8.3.4	Mixing with Spiders	469
8.3.5	Entanglement for Impure States	472

8.4	Measurements and Protocols with Spiders	476
8.4.1	ONB Measurements	476
8.4.2	Controlled Unitaries	479
8.4.3	Teleportation	482
8.4.4	Dense coding	485
8.4.5	Entanglement Swapping	486
8.4.6	Von Neumann Measurements	488
8.4.7	POVMs and Naimark Dilation	490
8.5	Summary: What to Remember	492
8.6	Advanced Material*	498
8.6.1	Spiders Are Frobenius Algebras*	498
8.6.2	Non-commutative Spiders*	502
8.6.3	Hairy Spiders*	505
8.6.4	Spiders as Words*	507
8.7	Historical Notes and References	507
9	Picturing Phases and Complementarity	510
9.1	Decorated Spiders	512
9.1.1	Unbiasedness and Phase States	512
9.1.2	Phase Spiders	517
9.1.3	Phase Spider Fusion	519
9.1.4	The Phase Group	522
9.1.5	Phase Gates	524
9.2	Multicoloured Spiders	529
9.2.1	Complementary Spiders	529
9.2.2	Complementarity and Unbiasedness	535
9.2.3	The CNOT-Gate from Complementarity	540
9.2.4	‘Colours’ of Classical Data	543
9.2.5	Complementary Measurements	545
9.2.6	Quantum Key Distribution	549
9.2.7	Teleportation with Complementary Measurements	552
9.3	Strong Complementarity	557
9.3.1	The Missing Rules	558
9.3.2	Monogamy of Strong Complementarity	561
9.3.3	Faces of Strong Complementarity	562
9.3.4	The Classical Subgroup	567
9.3.5	Parity Maps from Spiders	575
9.3.6	Classifying Strong Complementarity	578
9.4	ZX-Calculus	581
9.4.1	ZX-Diagrams Are Universal	582
9.4.2	ZX-Calculus for Clifford Diagrams	586
9.4.3	ZX for Dodos: Just Diagrams, Nothing Else	591

Preface

Glad you made it here! This book is about telling the story of quantum theory entirely in terms of pictures. Before we get into telling the story itself, it's worth saying a few words about how it came about. On the one hand, this is a very new story, in that it is closely tied to the past 10 years of research by us and our colleagues. On the other hand, one could say that it traces back some 80 years when the amazing John von Neumann denounced his own quantum formalism and embarked on a quest for something better. One could also say it began when Erwin Schrödinger addressed Albert Einstein's concerns about 'spooky action at a distance' by identifying the structure of composed systems (and in particular, their non-separability) as the beating heart of quantum theory.

From a complementary perspective, it traces back some 40 years when an undergraduate student named Roger Penrose noticed that pictures out-classed symbolic reasoning when working with the tensor calculus.

But 80 years ago the authors weren't around yet, at least not in human form, and 40 years ago there wasn't really that much of us either, so this preface will provide an egocentric take on the birth of this book. This also allows us to wholeheartedly acknowledge all of those without whom this book would never have existed (as well as some who nearly succeeded in killing it).

Things started out pretty badly for Bob, with a PhD in the 1990s on a then completely irrelevant topic of contextual 'hidden variable representations' of quantum theory – which recently have been diplomatically renamed to *ontological models* (Harrigan and Spekkens, 2010; Pusey et al., 2012). After a period of unemployment and a failed attempt to become a rock star, Bob ventured into the then even more irrelevant topic of von Neumann's quantum logic (Birkhoff and von Neumann, 1936) in the vicinity of the eccentric iconoclast Constantin Piron (1976).

It was there that category theory entered the picture, as well as serious considerations on the fundamental status of composition in quantum systems – something that went hand-in-hand with bringing quantum processes (rather than quantum states) to the forefront . . .

▲ In the case you are suffering from some kind of category theory phobia, do not stop reading here! Though it has influenced many of the ideas in this book, this is by no means a book about category theory!

... these considerations would ultimately provide the formal and the conceptual backbone for a pictorial approach to quantum theory. The categorical push in quantum foundations came initially from David Moore (1995), a very gifted researcher who suffered his academic end in the late 1990s, an era in which conceptually oriented physics suffered from widespread prohibition. In collaboration with Moore and Isar Stubbe, Bob made some early attempts towards a categorical reformulation of quantum theory (Coecke et al., 2001), which unfortunately inherited too many deficiencies from old-fashioned quantum logic. The main problem with quantum logic was its implicit assumption that the physical systems under consideration are always: ‘some part of the ostensibly external phenomenal world, supposed separated from its surroundings in the sense that its interactions with the environment can either be ignored or effectively modelled in a simple way’ (Moore, 1999). However, interactions with the environment happen to be something one should really care about!

After being kicked out of his university (cf. bureaucrats, village politics, and lots of hypocrisy), a second failed attempt in the arts, and looming unemployment, a bit of a miracle happened to Bob when two complete strangers, Prakash Panangaden and Samson Abramsky, arranged a ‘trial’ postdoc in the Computing Laboratory at Oxford, which was back then affectionately known as the Comlab. Despite knowing nothing about computer science and thinking of computer scientists as a bunch of nerds staring at screens all day, Bob found a home in this department and quickly discovered that, unlike quantum logicians, computer scientists had for a long time already studied the structure of interacting systems and were able to describe such systems elegantly in the language of category theory. In fact, in this particular computer science department, category theory was even taught at undergraduate level.

It was here that the second author entered the picture. While on a two-month exchange to Oxford from his homeland of Tulsa, Oklahoma (Fig. 0.1), Aleks happened to take the aforementioned undergraduate course in category theory, which was at the time taught by Samson. The mind-expanding nature of that course (including a guest lecture about weird-looking pictures of monoidal categories by an equally weird-looking guy) got Aleks interested enough to get involved in the subject. At Samson’s prompting, he started coming along to the group’s Quantum Lunch seminar. The seminar format consisted of a large pub lunch followed by a talk where a drunk and drowsy speaker addressed an equally drunk and drowsy audience on topics in the newborn subject of categorical quantum mechanics. It was great.

Two months turned into nine years, the Comlab became the ‘Department of Computer Science’, and though it seemed like nobody could remember when Aleks first starting hanging around, he ended up doing a master’s, PhD, and a postdoc.

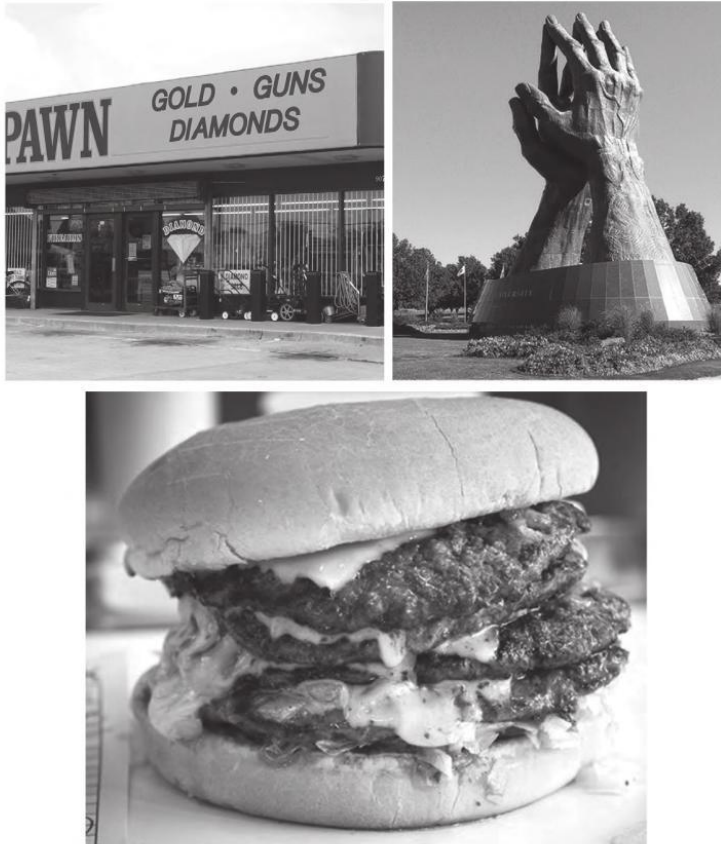


Figure 0.1 Some typical sights of Tulsa, Oklahoma.

Without the surprising wealth of both mathematical machinery and conceptual thinking in this unique computer science environment, this book simply would not have existed. In sharp contrast to the prohibition in the 1990s of the words ‘foundational’ and ‘conceptual’ in physics, in this new environment ‘foundational’ and ‘conceptual’ were (and still are) big virtues! This led to the birth of a new research community, in which computer scientists, pure mathematicians, philosophers, and researchers in the now-resurgent area of quantum foundations closely interact. It is probably even fair to say that this unique atmosphere contributed to the resurrection of the quantum foundations community as a whole, and along the way several of its highly respected practitioners have adopted the diagrammatic paradigm, notably Chiribella et al. (2010) and Hardy (2013a).

The conference series Quantum Physics and Logic (QPL), founded by Peter Selinger in 2003 under a different name (but with the same abbreviation!), was a particularly important forum for the development of the key results leading up to this book. In fact, the first paper about diagrammatic reasoning for novel quantum features (Coecke, 2003) was presented at the first QPL. The categorical formalisation of this result (Abramsky and

"THE FUNDAMENTALS OF CATEGORICAL QUANTUM MECHANICS"

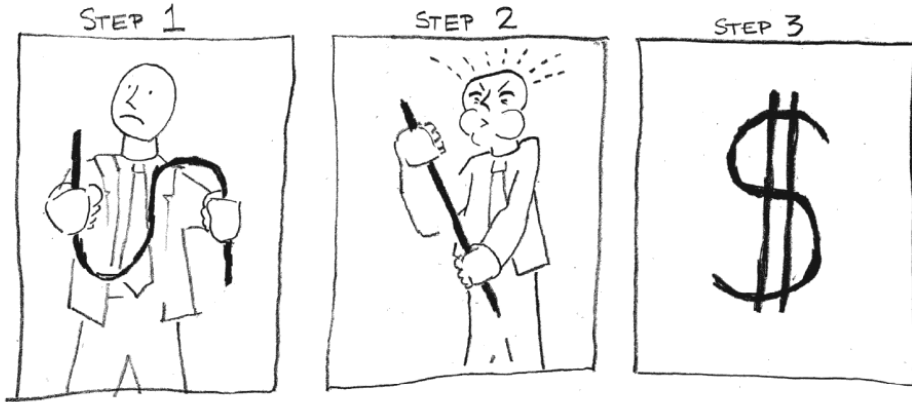


Figure 0.2 Yanking for cash.

Coecke, 2004), now referred to as categorical quantum mechanics, became a hit within the computer science semantics community, and ultimately allowed for several young people to establish research careers in this area. Top computer science conferences (e.g. LiCS and ICALP) indeed regularly accept papers on categorical quantum mechanics, and more recently leading physics journals (e.g. PRL and NJP) have started to do so too.

We are very grateful to have received a healthy flow of research funding (cf. Fig. 0.2), by the UK Engineering and Physical Sciences Research Council (EPSRC), the European Commission (FP6 FET Open), the US Office of Naval Research (ONR), the US Air Force Office of Scientific Research (AFOSR), the Foundational Questions Institute (FQXi), and the John Templeton Foundation (JTF). In particular, during the writing of this book both authors were generously supported by the latter. As a result, the Quantum Group at Oxford has grown over the past 10 years from 5 members in 2004 to 50 members now, and several of its former members have even started to build groups elsewhere, spreading the Gospel of pictures and processes. The constant interaction with the Quantum Group (and its numerous diaspora) has of course been absolutely essential in the development of this book.

So where did the idea of actually writing this book come from? Here at Oxford one typically doesn't like to mention 'that other place'. But, in the summer of 2012 Cambridge University Press contacted us to ask whether the diagrammatic language may be ready for a book. This generated the idea of teaching the Quantum Computer Science course that fall entirely with diagrams. The lecture notes would then provide the basis for a book, which would most certainly be ready by spring 2013. A quick glance at the first couple of pages in this book should tell you that this plan failed. The lecture notes were entirely dumped, and in fall 2013 we started again from scratch, leading up to what you are reading now.

Quite a number of things did happen from start to finish, like one of the authors relaunched a music career, met a girl, got married, made a baby, got a baby, and took a baby



Figure 0.3 Aleks' beard growth, as correlated with textbook completion.

to Beijing to watch him play a metal show. Meanwhile, the other author also got married, had a brief foray into stand-up comedy, got a position at Radboud University (amongst Bob's mortal enemies: the Dutch), and grew a humongous beard (Fig. 0.3). Both authors became well known at a local pub for being beaten up outside (rumour has it over a dispute on the interpretations of quantum theory, but neither author remembers too well). They also formed a southern country folk industrial noise band called the Quantum Dagger Orchestra.

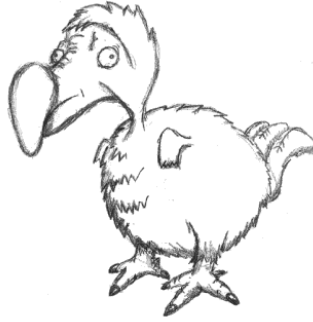
The students who have taken this course over the past several years have been an invaluable source of inspiration and practical guidance in terms of what works and what doesn't when teaching this drastically new approach to quantum theory. In particular, we would like to thank students Jiannan Zhang, William Dutton, Jacob Cole, Pak Choy, and Craig Hull in the class of 2013, Tomas Halgas in the class of 2014, and Ernesto Ocampo, Matthew Pickering, Callum Oakley, Ashok Menon, Ignacio Funke Prieto, and Benjamin Dawes in the class of 2015, who have all contributed to this finished product by pointing out typos in those original (and revised) lecture notes.

We would especially like to thank Yaared Al-Mehairi, Daniel Marsden, Katie Casey, John-Mark Allen, Fabrizio Genovese, Maaïke Zwart, Hector Miller-Bakewell, Joe Bolt, John van de Wetering, and Adrià Garriga Alonso, who all provided substantial qualities and/or quantities of corrections, and the class tutors Miriam Backens, Vladimir Zamdchiev, Will Zeng, John-Mark Allen, and Ciaran Lee for their valuable overall input, model solutions, and for bearing the brunt of the students' frustration when some of the more 'experimental' exercises went wrong.

Detailed corrections on the final manuscript were provided by booze brothers John Harding and Frank Valckenborgh, and the pictures of angry Bob were all taken by yet another booze brother, Ross Duncan. Consistent reminders for Aleks to eat and avoid head trauma were graciously provided by his wife Claire. Bob failed to take account of these reminders by his wife Selma.

All of the diagrams in this book were created using PGF/TikZ package for LaTeX and the TikZiT software. Grab the latest version from tikzit.github.io.

Finally: Why Cambridge University Press and not Oxford University Press? Because a CUP causes a whole lot of magic in this book while an OUP is ... no clue.



He's a dodo. Not your typical run-of-the-mill dodo, but a *quantum dodo*. We will assume that Dave behaves in the same manner as the smallest non-trivial quantum system, a two-level system, which these days gets referred to as a quantum bit, or *qubit*. Let's compare Dave's state to the state of his classical counterpart, the *bit*. Bits form the building blocks of classical computers, whereas (we will see that) qubits form the building blocks of quantum computers. A bit:

1. admits two states, which we tend to label 0 and 1,
2. can be subjected to any function, and
3. can be freely read.

Here, 'can be subjected to any function' means that we can apply any function on a bit to change its state. For example, we can apply the 'NOT' function to a bit, which interchanges the states 0 and 1, or the 'constant 0' function which sends any state to 0. What we mean by 'can be freely read' is that we can read the state of any bit in a computer's memory without any kind of obstruction and without changing that state.

The fact that we even mention all of this may sound a bit odd...until we compare this to the quantum analogue. A qubit:

1. admits an entire sphere of states,
2. can only be subjected to rotations of the sphere, and
3. can only be accessed by special processes called *quantum measurements*, which only provide limited access, and are moreover extremely invasive.

The set of states a system can occupy is called the *state space* of that system. For classical bits, this state space contains just two states, whereas a qubit can be in infinitely-many states, which we can visualise as a sphere. In the context of quantum theory, this state space is called the *Bloch sphere*. For the sake of explanation, any sphere will do, so we'll just take the Earth. There's plenty of space on Earth for two states of a bit, so put 0 on the North Pole and 1 on the South Pole:



The particular choice of North Pole/South Pole is not important, but it is important that they are *antipodal* points on the sphere.

Since we can only apply rotations to the sphere of qubit-states, we cannot map both 0 and 1 to 0 (as we could with classical bits), simply because there is no rotation that does that. On the other hand, there are lots of ways to interchange 0 and 1, since there are many (different!) rotations that will turn a sphere upside-down.

So what are quantum measurements? Just like when we read a normal bit, measuring a qubit will produce one of two answers (e.g. 0 or 1, hence the name qubit). However, this act of ‘measuring’ is not quite as innocent as simply reading a bit to get its value. To get a feel for this, we return to Dave. Since qubits can live anywhere in the world, Dave – like one particularly famous (classical) dodo – lives in Oxford:



Now, suppose we wish to ascertain where in the world certain animals live, subject to the following assumptions:

1. we are only allowed to ask whether an animal lives at a specific location on Earth or its antipodal location;
2. all animals can talk and will always answer ‘correctly’; and
3. predatory animals will refrain from eating the questioner.

If we ask a polar bear whether she lives at the North Pole or the South Pole, then she'll say 'the North Pole'. If we ask again, she'll say 'the North Pole' again, because that's just where polar bears are from. Similarly, if we ask a penguin, he'll keep saying 'the South Pole', as long as we keep asking.

On the other hand, what will Dave say if we ask him whether he lives at the North Pole or the South Pole? Now, Dave doesn't really understand the question, but since dodos are a bit thick, he'll give an answer anyway. However, assumption 2 was that all animals will answer correctly. Consequently, as soon as Dave says 'the North Pole', his statement is correct: he actually is at the North Pole!



Now, if we ask him again, he'll say 'the North Pole' again, and he'll keep answering thus until he's eaten by a polar bear (Fig. 1.1). Alternatively, if he had initially said 'the South Pole', he would immediately have been at the South Pole.





Figure 1.1 A polar bear attempting a 'demolition measurement' on Dave.

So, no matter what answer Dave gives, his state has changed. The fact that he was originally in Oxford is permanently lost. This phenomenon, known as the *collapse* of the quantum state, happens for almost all questions (i.e. measurements) we might perform. Crucially, this collapse is almost always *non-deterministic*. We almost never know until we measure Dave whether he'll be at the North Pole or the South Pole. We say 'almost', because there is one exception: if we ask whether Dave is in Oxford or the Antipodes Islands, he'll say 'Oxford' and stay put.

While quantum theory cannot predict with certainty the fate of Dave, what it does provide are the *probabilities* for Dave to either collapse to the North Pole or to the South Pole. In this case, quantum theory will tell us that Dave is more likely to go to the North Pole and get eaten by a polar bear than to go to the South Pole and chill with some penguins. The dodo is extinct for a reason after all ...

1.2 So What's New?

Almost a century has passed since Dave's unfortunate travels to the North Pole. In particular, the past two decades have seen a humongous surge in new kinds of research surrounding quantum theory, ranging from re-considering basic concepts (Fig. 1.2) to envisioning radically new technologies. A paradigmatic example is *quantum teleportation*, whereby the non-local features of quantum theory are exploited to send a quantum state across (sometimes) great distances, using nothing but a little bit (actually two little bits ...) of classical communication. Quantum teleportation exposes a delicate interaction between quantum theory and the structure of spacetime at the most fundamental level. At the same time, it is also a template for an important quantum computational model (measurement-based quantum computing), as well as a component in many quantum communication protocols.

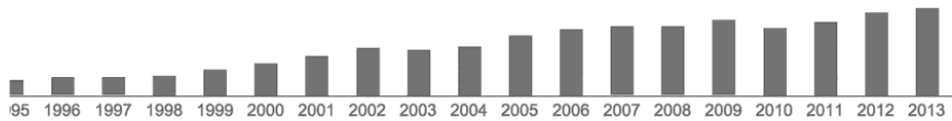


Figure 1.2 The paper by Einstein, Podolsky, and Rosen, which was the first to identify quantum non-locality, has enjoyed a huge surge in citations over the past two decades according to Google Scholar, now making it Albert Einstein’s most cited paper. And considering the competition, that’s saying something.

Quantum theory as we now know it – that is to say, its formulation in terms of *Hilbert spaces* – first saw daylight in 1932 with John von Neumann’s book *Mathematische Grundlagen der Quantenmechanik*. On the other hand, quantum teleportation was only discovered in 1992. Hence the question:

Why did it take 60 years for quantum teleportation to be discovered?

A first explanation is that within the tradition of physics research during those 60 years, the question of whether something like quantum teleportation would be possible was simply never asked. It only became apparent when researchers stepped outside the existing scientific tradition and asked a seemingly bizarre question:

What are the information processing features of quantum theory?

However, one could go a step further and ask why it was even necessary to first pose such a question for teleportation to be discovered. Why wasn’t it plainly obvious that quantum theory allowed for quantum teleportation, in the same way that it is plainly obvious that hammers are capable of hitting nails? Our answer to this question is that the traditional language of Hilbert spaces just isn’t very good at exposing many of the features of quantum theory, and in particular, those features such as teleportation that involve the interaction of multiple systems across time and space. Thus, we pose a new question:

What is the most appropriate language to reason about quantum theory?

The answer to this question is what this book is all about. The reader will learn about many important new quantum features that rose to prominence within the emerging fields of quantum computation, quantum information, and quantum technologies, and how these developments went hand-in-hand with a revival of research into the foundations of quantum theory. All of this will be done by using a novel presentation of quantum theory in a purely diagrammatic manner. This not only consists of developing a two-dimensional notation for describing and reasoning about quantum processes, but also of a unique methodology that treats quantum processes, and most importantly *compositions* of processes, as first-class citizens.

1.2.1 A New Attitude to Quantum Theory: ‘Features’

Since its inception, many prominent thinkers were deeply unsettled by quantum theory. A great deal of effort and ingenious mathematics in the early twentieth century went

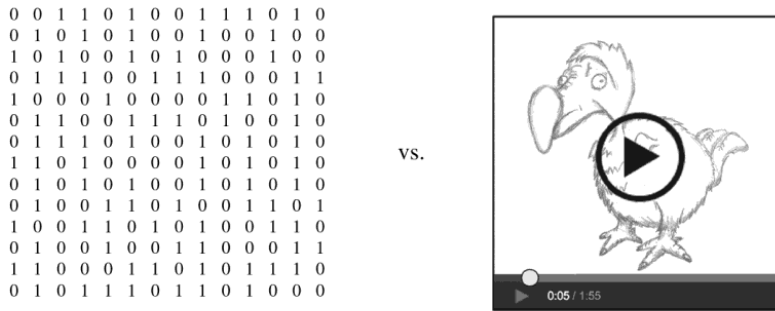


Figure 1.5 Contrasting a low-level and a high-level representation of the digital data that one may find within a computational device.

1.2.2 A New Form of Mathematics: ‘Diagrams’

It should be emphasised that discovering these new quantum features wasn’t trivial and involved some very smart people. Our bold claim is that when one adopts the appropriate language for quantum theory, these features jump right off the page. Conversely, the traditional, Hilbert space–based language of quantum theory forms a major obstruction to discovering such features. To give some idea of why this is the case, we will make use of some simple metaphors.

Imagine that you were trying to determine what was happening in a video just by looking at its digital encoding (Fig. 1.5). Obviously this is a more or less impossible task. While digital data, i.e. strings of 0s and 1s, is the workhorse of digital technology, and while it is possible to understand ‘in principle’ how they encode all of the media stored on your hard drive, asking a person to decode a particular string of binary by hand is more suitable for punishing greedy bankers and corrupt politicians than solving interesting problems.

Of course, even skilled computer programmers wouldn’t be expected to interact directly with binary data. Somewhere along the way to modern computer programming came the advent of assembly language, which gives a (somewhat) human-readable translation for individual instructions sent to a computer processor. While this made it more practical to write programs to drive computers, it still takes a lot of head-scratching to figure out what any particular piece of assembly code does. Using *low-level languages* such as assembly language creates an artificial barrier between programs and the concepts that they represent and places practical limits on the complexity of problems those programs can solve. For this reason, virtually every programmer today uses *high-level languages* in their day-to-day work (Fig. 1.6).

Similarly, ‘detecting new quantum features’ in terms of the traditional (i.e. low-level) language for quantum theory, namely ‘strings of complex numbers’ (rather than ‘strings of 0s and 1s’), isn’t that easy either. This could explain why it took six highly esteemed researchers to discover quantum teleportation, some 60 years since the actual birth of the quantum theoretical formalism. By contrast, the diagrammatic language we use in this book is a *high-level* language for exploring quantum features (Fig. 1.7). We will soon see

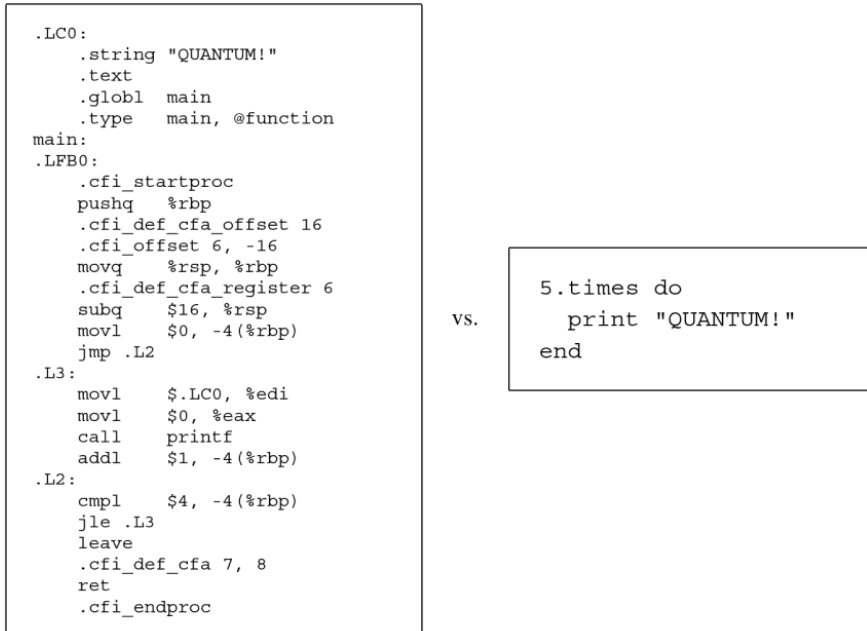


Figure 1.6 Contrasting a low-level and a high-level language for computer programs. The programs on the left and right perform the same task, but one is written in the low-level x86 assembly language and one in the high-level language Ruby.

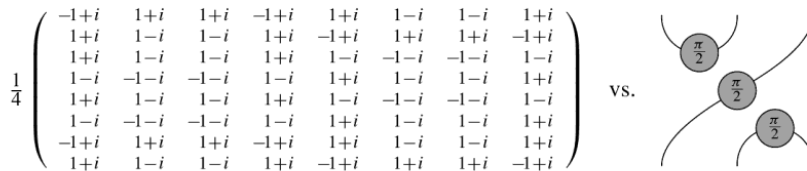


Figure 1.7 Contrasting a low-level and a high-level language for quantum processes, just like we contrasted the low-level and a high-level representation for digital data in Fig. 1.5 and a low-level and a high-level programming language in Fig. 1.6.

that by embracing the diagrammatic language for quantum theory, features like quantum teleportation are pretty much staring you in the face!

Although it goes beyond the scope of this book, it is worth mentioning that the diagrammatic language we use has found applications in other areas as well, such as modelling meaning in natural language (Fig. 1.8), doing proofs in formal logic, control theory, and modelling electrical circuits.

Diagrams are also becoming increasingly important in some fancy research areas of pure mathematics, such as knot theory, representation theory, and algebraic topology. By using diagrams we eliminate a huge amount of redundant syntactic garbage in representing

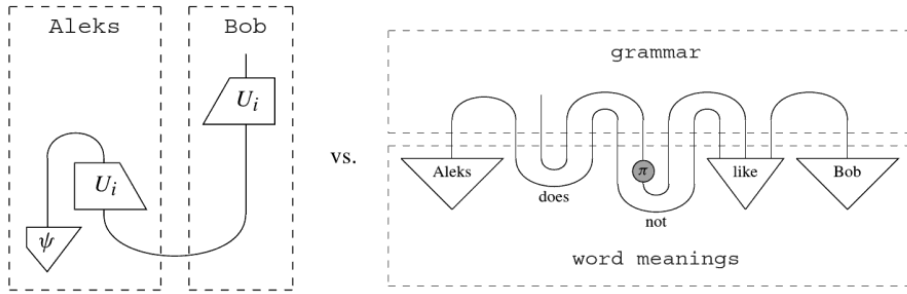


Figure 1.8 Comparing diagrammatic representations of quantum processes to those of ‘the flow of meaning’ in natural language. While these are two very different contexts, Aleks and Bob feel well at home in both due to their diagrammatic similarity. In the diagram representing natural language, the upper half represents the grammatical structure, while the bottom half represents meaning of individual words, and the overall wiring exposes how the meanings of these words interact in order to produce the meaning of the entire sentence.

$$(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2) \quad \text{vs.} \quad \begin{array}{c} \boxed{g_1} \quad \boxed{g_2} \\ | \quad | \\ \boxed{f_1} \quad \boxed{f_2} \end{array}$$

Figure 1.9 Two distinct syntactic descriptions corresponding to the same diagram. In terms of the symbolic language that is used on the left, two syntactically non-equal expressions might mean the same thing. On the other hand, in terms of the graphical language that is used on the right, there is only one representation. This example is explained in great detail in Section 3.2.4.

mathematical objects (Fig. 1.9), freeing us to concentrate on the important features of the mathematical objects themselves.

There are clear indications that diagrammatic reasoning will become increasingly important in the sciences in general, and this book represents the first attempt to comprehensively introduce a big subject like quantum theory entirely in this new language. By reading this book – or even more, taking a course based on this book – you, like the monkeys launched into space in the 1960s, are the ‘early adopters’ (a.k.a. ‘test subjects’) in a totally new enterprise.

1.2.3 A New Foundation for Physics: ‘Process Theories’

By taking diagrammatic language as a formal backbone for describing quantum theory (or any other physical theory, for that matter) one also subscribes to a new perspective on physical theories.

First, traditional physical theories take the notion of a ‘state of a system’ as the primary focus, whereas in diagrammatic theories, it is natural to treat arbitrary processes on equal footing with states. States are then treated just as a special kind of process, a

‘preparation’ process. In other words, there is a shift from focussing on ‘what is’ to ‘what happens’, which is clearly a lot more fun. This is very much in line with the concerns of computer science, where the majority of time and energy goes into reasoning about processes (i.e. programs), and states (i.e. data) only exist to be used and communicated by programs. It is also becoming clear that one should focus not just on single programs but on collections of *interacting* programs to understand the complex, distributed computer systems that are becoming increasingly prevalent in the modern world.

Another example where studying interaction is crucial to understanding a system comes from biology. While one can (in principle) deduce the coat of an animal from its genetic code, this does not explain why that animal has such a coat. On the other hand, if we look at where an animal lives or how it attracts a mate, for example, this can immediately become clear. Similarly, rather than concentrating on systems in isolation, our approach to physics looks at the overall structure of many systems and processes and how they compose. We call such a structure consisting of all the ‘allowed processes’ and how these interact a *process theory*.

Schrödinger realised early on that the most startlingly non-classical features of quantum theory came from looking not at a single system, but rather at how multiple systems behave together. Rather than acting as a collection of individuals, quantum systems establish complex relationships, and it is these relationships that suddenly enable amazing new things:

When two systems, of which we know the states by their respective representatives, enter into temporary physical interaction due to known forces between them, and when after a time of mutual influence the systems separate again, then they can no longer be described in the same way as before, viz. by endowing each of them with a representative of its own. I would not call that *one* but rather *the* characteristic trait of quantum mechanics, the one that enforces its entire departure from classical lines of thought.

Schrödinger says that the most important trait of quantum mechanics only becomes apparent when we study the interactions of two systems. So, one might expect any approach to quantum theory to start from a point of view that emphasises compositionality from page 1. But oddly, if you pick up a random textbook on quantum theory, this is not the point of view you will see. It was not until the late 1990s – largely prompted by the discoveries we discussed in Section 1.2.1 – that this idea made it back into the mainstream.

The concept of a process theory does of course put composition at the forefront, and it suggests a natural way of reasoning about processes. One should pare down the nitty-gritty aspects of how processes are mathematically defined and seek out high-level principles that govern their interactions. These principles taken together constitute what can be called the *logic of interaction* for a process theory. Von Neumann also thought that quantum theory should be understood in terms of logical principles. Three years after he published *Mathematische Grundlagen*, von Neumann wrote:

I would like to make a confession which may seem immoral: I do not believe absolutely in Hilbert space no more. [sic]

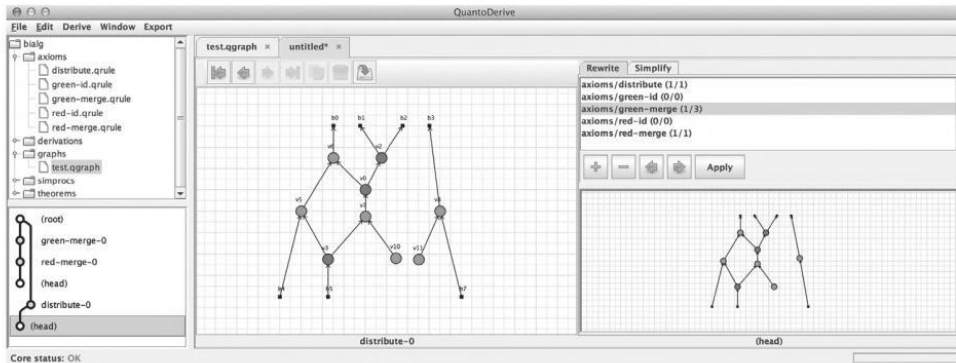


Figure 1.10 Quantomatic: a diagrammatic proof assistant.

He went on to say that it is not the Hilbert-space structure of quantum theory that is physically relevant, but rather the quantum analogue of ‘logical propositions’, namely those properties verifiable by means of quantum measurements. However, this new kind of logic, called *quantum logic*, ultimately failed to replace Hilbert space as a conceptual basis for quantum theory. Its biggest stumbling block was its complete focus on systems in isolation and its inability to obtain any conceptual account of composed systems. More pragmatically, the passage from Hilbert space to quantum logic seems to make it more difficult to establish new facts or discover new features of quantum theory, typically requiring extreme cleverness on the part of its practitioners to establish even basic facts.

In contrast, this new kind of interaction logic via process theories has very quickly become a practical tool for high-level reasoning about quantum systems and beyond, not in the least due to its intuitive diagrammatic language. It has even formed the basis of a diagrammatic *proof assistant* – i.e. an interactive software tool that constructs proofs (semi-)automatically – called Quantomatic (Fig. 1.10).

So, what should we call this new way of reasoning about quantum theory entirely with diagrams, focussing crucially on its processes and the logic of their interactions? Since the term ‘quantum logic’ is already trademarked, we’ll have to use something a bit more descriptive ...

1.2.4 A New Paradigm: ‘Quantum Pictorialism’

In Section 1.2.1 we said that the dissatisfaction with quantum theory obstructed people from realising what the actual features of quantum theory were and how those features could be put to good use. Then, by asking the right ‘positive’ question, many new features were discovered. We went on to argue that in an adequate mathematical language, the features of quantum theory should be plainly obvious. Taking things one step further, one could wonder if this adequate mathematical language isn’t just easier to work with, but is also closer to what the world is made up of!

The Holy Grail of theoretical physics is to come up with a theory of quantum gravity. It is likely that to develop a consistent theory of quantum gravity, some of the core assumptions

exactly as predicted by quantum theory. It should be mentioned that there were a handful of objections to that initial experiment, in the form of ‘loopholes’ in that particular demonstration of non-locality. All of these have meanwhile been closed by other experiments (Weihs et al., 1998; Rowe et al., 2001; Hensen et al., 2015). On top of that, the huge variety of experiments that have been done confirming some form or another of non-locality for quantum theory is pretty compelling, to say the least (see e.g. Rauch et al., 1975; Zeilinger, 1999; Pan et al., 2000; Gröblacher et al., 2007).

Another development prompted by the EPR paper was the quest for interpretations of quantum theory. Given EPR’s claim of incompleteness of quantum theory, one family of interpretations was all about completing quantum theory, although due to Bell’s theorem, any such an attempt is bound to be non-local. The most famous one of these is the hidden variable interpretation due to David Bohm (1952a,b). Another one loved by Hollywood is the many-worlds interpretation due to Hugh Everett III (1957). The official default interpretation of quantum theory is the Copenhagen interpretation due to Niels Bohr and Werner Heisenberg, which in the eyes of many is a non-interpretation, given that at first sight it provides nothing more than a recipe to compute probabilities. A detailed survey and extensive discussion of the interpretations of quantum theory can be found in Bub (1999).

The shut-up-and-calculate slogan is often associated with Richard Feynman, who did in many ways embody this way of working, but in fact it was coined by David Mermin (May 2004), who very much did not skirt around foundational questions in quantum theory. He used the term not to refer to this common practice in particle physics, but rather to give his view on the Copenhagen interpretation (Mermin, April 1989).

While on the topic of Feynman, it is worth mentioning that he was the first to realise that there was something quantum systems are really good at: simulating themselves (Feynman, 1982). Thus, his notion of a quantum simulator contained the first seeds of the idea of quantum computation. The discovery of less self-referential applications for quantum features in information processing began a few years later with the advent of quantum key distribution (Bennett and Brassard, 1984). A year later, at the University of Oxford, David Deutsch (1985) gave a formulation for a universal quantum computer, the quantum analogue to Turing’s universal machine (Turing, 1937). This led to the discovery of quantum algorithms that substantially outperformed any classical algorithm (Deutsch and Jozsa, 1992; Shor, 1994; Grover, 1996; Simon, 1997). The term ‘qubit’ was coined by Schumacher (1995).

Quantum teleportation was proposed by Bennett et al. (1993), and its first experimental realisation was by Bouwmeester et al. (1997). The question of why it took 60 years for quantum teleportation to be discovered was asked by one of the authors in a seminar at the Perimeter Institute of Theoretical Physics and was immediately answered by Gilles Brassard, co-inventor of quantum teleportation and a pioneer of the quantum information endeavour as a whole, who happened to be in the audience. He said that no one before had considered the information-processing features of quantum theory and had therefore simply not thought to ask the question. This exchange is reported in Coecke (2005).

The diagrams used in this book are an extension of those used by Roger Penrose (1971), who introduced them as an alternative for ordinary tensor notation (see Section 3.6.1).

However, many similar diagrammatic languages were invented prior to this or reinvented later.

In programming language theory, flow charts were among the first abstract presentations of programs and algorithms. These flow charts, introduced in Gilbreth and Gilbreth (1922) under the name ‘process charts’, are widely used in many other disciplines too. In quantum information the use of diagrammatic representations started with quantum circuits, a notation borrowed from circuits made up of Boolean logic gates, to which a number of new properly quantum gates were added (see e.g. Nielsen and Chuang, 2010).

A diagrammatic notation specifically tailored towards the processes responsible for quantum weirdness was first introduced in Coecke (2003, 2014a) and, independently, also in Kauffman (2005). These diagrams were provided with an axiomatic underpinning in Abramsky and Coecke (2004), and independently in Baez (2006), paving the way to a diagrammatic approach for quantum theory as a whole. The main pillars supporting the story outlined in this book are the diagrammatic representation of mixed states and completely positive maps by Selinger (2007), the diagrammatic representation of classical data as ‘spiders’ in (Coecke and Pavlovic, 2007; Coecke et al., 2010a), the diagrammatic representation of phases, complementarity, and the introduction of strong complementarity by Coecke and Duncan (2008, 2011), again in terms of spiders, and the causality postulate introduced by Chiribella et al. (2010). ‘Quantum pictorialism’ was coined in Coecke (2009).

Important topics in the area of quantum computing and related areas that were diagrammatically explored are quantum circuits (Coecke and Duncan, 2008), (topological) measurement-based quantum computing (Coecke and Duncan, 2008; Duncan and Perdrix, 2010; Horsman, 2011), quantum error correction (Duncan and Lucas, 2013), quantum key exchange (Coecke and Perdrix, 2010; Coecke et al., 2011a), non-locality (Coecke et al., 2011b, 2012), and quantum algorithms (Vicary, 2013; Zeng and Vicary, 2014).

Structural theorems about quantum theory emerging from the diagrammatic approach include a number of completeness theorems (Selinger, 2011a; Duncan and Perdrix, 2013; Backens, 2014a; Kissinger, 2014b) and some representation theorems (Kissinger, 2012a; Coecke et al., 2013c).

For applications of the kinds of diagrams that we consider here to other scientific disciplines, there are, for example, Coecke et al. (2010c) and Sadzadeh et al. (2013) for applications to natural language, Mellies (2012) for logic in computer science, Pavlovic (2013) for computability, Hinze and Marsden (2016) for programming, Baez and Fong (2015) for applications to electrical circuits, Bonchi et al. (2014a) and Baez and Erbele (n.d.) for applications in control theory, Hedges et al. (2016) for applications in economic game theory, Baez and Lauda (2011) for a prehistory, Baez and Stay (2011) for a Rosetta Stone, and Coecke (2013) for an alternative Gospel.

A discussion of von Neumann’s discontent with Hilbert space is in Redei (1996), from which we also took the second quote in Section 1.2.3. Attempted modifications/generalisations/axiomatisations of quantum theory, all to a great extent inspired by quantum logic (Birkhoff and von Neumann, 1936), were pioneered by Mackey (1963), Jauch (1968), Foulis and Randall (1972), Piron (1976), and Ludwig (1985). Coecke et al. (2000) provides

a survey of these approaches. A tutorial on how property lattices yield Hilbert spaces is in Stubbe and van Steirteghem (2007). A survey of Foulis and Randall's 'test space' formalism (or 'manuals' formalism) is in Wilce (2000). Ludwig's approach has recently become very prominent again under the name 'generalised probabilistic theories' (Barrett, 2007). Also, several researchers have tried to combine the earlier axiomatic approaches with diagrams and/or compositional structure (Harding, 2009; Heunen and Jacobs, 2010; Jacobs, 2010; Vicary, 2011; Abramsky and Heunen, 2012; Coecke et al., 2013a,b; Tull, 2016).

Giving processes a privileged role in quantum theory was already present in the work of Whitehead (1957) (cf. the quotation at the beginning of Chapter 6) and in Bohr (1961) and became more prominent in Bohm (1986). Process ontologies trace back to the pre-Socratics, most notably to Heraclitus of Ephesus in the sixth century BC (cf. the quotation at the beginning of Chapter 2). Diagrams, and hence a privileged role for processes and composition thereof, are used as a canvas for drafting theories of physics in Chiribella et al. (2010), Coecke (2011), and Hardy (2011, 2013b).

The second quote in Section 1.2.3 on the importance of the role of composition in quantum theory is taken from Schrödinger (1935). The first proper 'interaction logic' was *Geometry of Interaction* due to Jean-Yves Girard (1989), which was recast in a form more resembling the language used in this book in Abramsky and Jagadeesan (1994), and even more so in Duncan (2006). That quantum picturalism can be seen as a logic of interaction is argued in Coecke (2016), on the basis that the roots of logic are language and that the use of logic is artificial reasoning. The diagrammatic proof assistant Quantomatic is described in (Kissinger and Zamdzhiev, 2015) and at the time of this writing is still being actively developed. It is available from the project Website quantomatic.github.io.

2

Guide to Reading This Textbook

Nothing endures but change.

– *Heraclitus of Ephesus, 535–475 BC*

2.1 Who Are You and What Do You Want?

While there is already a plethora of textbooks on quantum theory and its features, this one is unique because it is based on quantum picturalism.

Prerequisites. There are hardly any prerequisites to this textbook. We do not expect our readers to have a background in physics or computer science or to have any profound background in mathematics. In principle, some basic secondary-school mathematics should be sufficient.

For example, linear algebra (and of course quantum theory) is presented from scratch in a diagrammatic manner. However, this does not mean that the first half of this book will be a boring read for the specialist, since these presentations from scratch are radically different from the usual ones.

Target audience. Given its low entrance fee, as well as its unique form and content, this textbook should appeal to a broad audience, ranging from students to experts from a wide range of disciplines including physicists, computer scientists, mathematicians, logicians, philosophers of science, and researchers from other areas with a multidisciplinary interest, such as biologists, engineers, cognitive scientists, and educational scientists.

One particular target audience for this book consists of students and researchers in quantum computation and quantum information, as we will apply the tools from quantum picturalism directly to these areas. A practising quantum computing researcher may discover a new set of tools to attack open problems where traditional methods have failed, and a student may find some subjects explained in a manner that is much easier to grasp.

Another target audience consists of students and experts with an interest in foundations and/or philosophy of physics, who can read in this book about a process-oriented approach to physics that takes composition of systems as a first-class citizen, rather than a derived notion. In particular, this is the first book that uses diagrammatic language to capture the

idea of a process theory and puts such process theories forward as a new foundation for quantum theory, in which all standard quantum theoretical notions can be expressed.

Yet another target audience consists of logicians and computer scientists, who may want to learn about diagrams as a new kind of logical paradigm, which emphasises ‘composition’ over ‘proposition’. For them, learning quantum theory may just be an added bonus to learning about this new paradigm for theory development.

But since this textbook covers the essential ingredients of a standard textbook on quantum computation and quantum foundations, it can just as well be used as a first introduction to those fields. Even though our notation is different from the one used in other textbooks, we cover the core curriculum of a first course in quantum computation and make a continual effort to relate the concepts and notations we introduce to those used more commonly in the literature.

Similarly, this textbook can be used as a first introduction to diagrammatic reasoning, being pretty much the first textbook that does that too.

2.2 The Menu

While we cannot yet offer dodo steaks, with recent advances in science, it may not be too long before pigeons give birth to new Daves and Davettes.

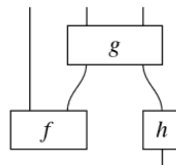
2.2.1 How Diagrams Evolve in This Book

In this book two stories more or less evolve in parallel:

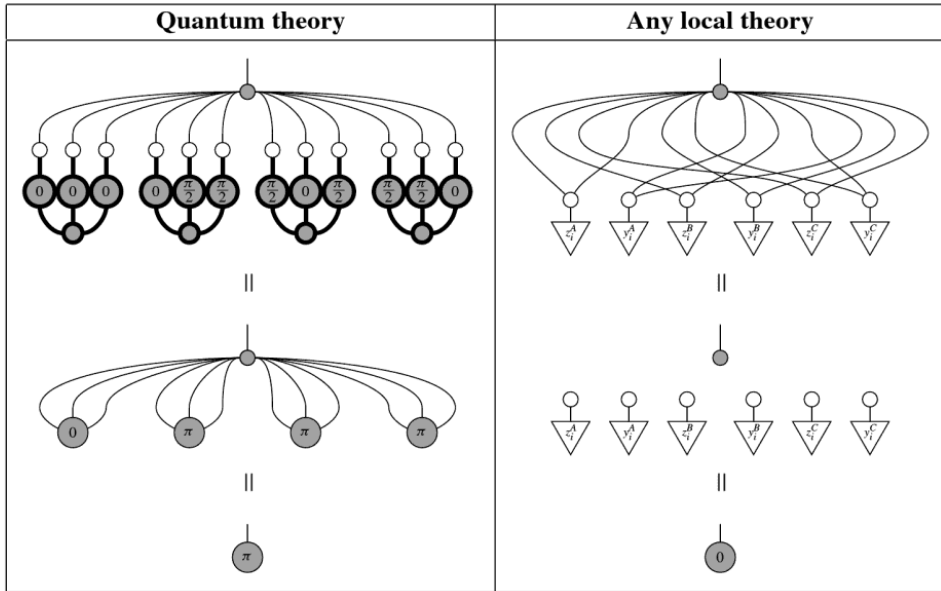
- the development of the diagrammatic language and
- the presentation of quantum theory as a process theory.

Quantum picturalism is indeed all about how these two are closely intertwined. We begin with a very general diagrammatic language and gradually add features to increase its expressiveness. Thus, we start with a language that is general enough to describe many different kinds of processes and gradually home in on quantum processes. Along the way, we present quantum features as and when the language is rich enough to discuss them. As a result, we will encounter features such as quantum teleportation way before more concrete notions such as qubits. All together there are five major jumps in the expressiveness of diagrams on the way to capturing full-blown quantum theory.

1. We first introduce a very basic diagrammatic language in Chapter 3 consisting of nothing but *boxes* and *wires*. This gives us a natural way to express compositions of processes in any process theory:



While the account of non-locality in standard textbooks involves pages mixing up words and formulas, for us it simply boils down to two diagrammatic computations, one about quantum theory and one about local theories, which yield contradictory results:

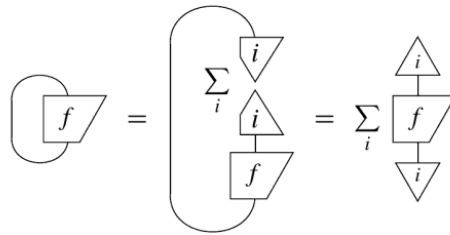


We will learn that the diagram on the left models four measurements performed on a GHZ-state, followed by computing the *parity* of the measurement outcomes. In other words, our theory is telling us whether to expect an even or an odd number of clicks (or beeps, flashes, etc.) coming from our measurement devices. We will see that the reduction to π on the left says that, according to quantum theory, the parity will be odd. Compare this with the derivation on the right, which assumes that there are some pre-established correlations that determine the measurement outcomes. This is always the case with a local theory, since all correlations of distant events can be traced back to some common cause. The reduction to 0 on the right says that any local theory predicts an even number of clicks, and hence there is a contradiction between quantum theory and locality.

This example is taken from Chapter 11, ‘Quantum Foundations’, in which we also present a toy theory that looks very much like qubit quantum theory, but fails to be non-local. There are two more themed chapters: Chapter 12, ‘Quantum Computation’, where we address standard topics such as the circuit model of quantum computation and quantum algorithms, as well as less standard (but increasingly important) topics such as measurement-based quantum computation. In Chapter 6, ‘Quantum Resources’, we present a general framework to study resources in quantum theory, with quantum entanglement as a particularly important example. We also show how qualitatively distinct types of quantum entanglement can be thought of as very differently behaving spiders.

2.2.3 Some Intermediate Symbolic Pollution

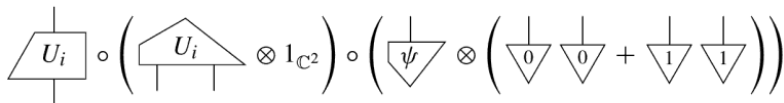
We haven't mentioned Chapter 5 yet. This chapter doesn't contribute to the development of quantum picturalism, but makes the connection with the usual quantum theoretical formalism. The main question addressed is the following. Given a process theory with string diagrams, when do wires represent Hilbert spaces and boxes represent linear maps? As an answer to this question, we adjoin some symbols to string diagrams, resulting in a hybrid diagram–symbol formalism in which one encounters computations like this one:



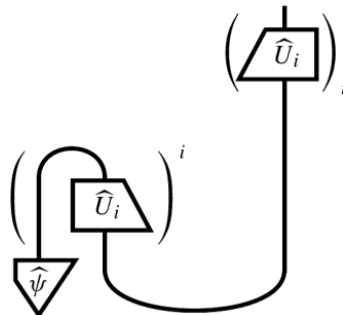
Such a hybrid formalism is in itself useful, and is widely used in areas of mathematics such as knot theory. More importantly, it introduces the usual formalism to those who don't know it already, and for those who are familiar with it, it makes clear how the diagrams relate to it. Finally, and perhaps most importantly, it allows us to state what exactly one can compute with string diagrams.

Something this allows us to do, which cannot be found in standard textbooks, is to make a smooth transition from linear maps to quantum processes and ultimately to processes that model quantum non-determinism and the classical-quantum interaction in a purely graphical way.

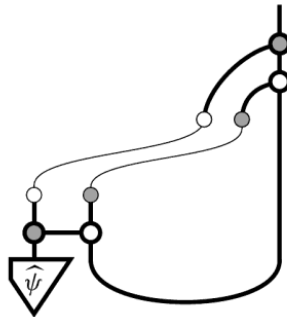
Along the way, we, like Dante, will proceed from linear algebraic Hell:



through Purgatory:



and ultimately to a purely diagrammatic Paradise:



2.2.4 Summaries, Historical Notes, References, Epigraphs

Each chapter (with the exception of Chapter 10, which is already a summary) contains a short section entitled ‘Summary: What to Remember’ listing the essential material to be taken from it.

At the end of each chapter there is also a short section entitled ‘Historical Notes and References’ in which we sketch the historical development of the material covered in that chapter, list some key references, and suggest some further reading.

The epigraphs appearing at the beginning of each chapter are all relevant to the contents of that chapter. For some it will be immediately clear from the text. Determining the relevance of the others is left as an exercise for the reader.

2.2.5 Starred Headings and Advanced Material Sections

Any sections, theorems, remarks, examples, or exercises that have a star (*) as superscript, e.g. ‘Remark* x.y.z’, are to be considered as optional. Typically they require some knowledge that only a fraction of our readers would either know about or be interested in, and hence they are only intended for that particular fraction of readers. For instance, a starred remark may require knowledge of some advanced concepts from linear algebra, quantum theory, or programming. Notably, each chapter has a section entitled ‘Advanced Material’, which contains material that particularly advanced students or specialists may find interesting. These in particular include clarification of the connection between diagrams and monoidal categories. Some attention is also given to currently ongoing research in quantum pictorialism, how it is related to recent developments in pure mathematics, and the surprising connection with natural language meaning.

2.3 FAQ

Over the years, we have noticed that people ask a few questions all of the time. We also anticipate a couple of new questions arising about this book in particular. We’ll try to address both of these here.

Q1: *Why does it take X pages to get to some basic stuff such as Y ?*

A: There are a few reasons for this:

- As the title suggests, this is a first course not only in quantum theory, but also in diagrammatic reasoning. Thus, we introduce features at the exact points where we have a rich enough language to talk about them. This doesn't always happen in the order you might expect.
- We assume no preliminaries and construct as much as possible from first principles, anticipating a very broad spectrum of potential readers. This means introducing many things, such as linear algebra, that will be old hat for many readers. However, we develop these basic concepts in such a drastically different, diagrammatic way that we think there is something for everyone in each chapter.
- It turns out that diagrams take up a lot of space. Sorry, trees.

Q2: *Where's the beef (i.e. the numbers)?*

A: A traditionally held belief is that the predictive content of a physical theory lies in its ability to produce numbers, such as probabilities. Many find it difficult to reconcile this idea with a diagram, which seems like a discrete, logical object. However, we will see in Section 3.4.1 that numbers arise naturally as special kinds of diagrams. That being said, the most interesting features we will highlight in this book are qualitative, not quantitative. As previewed in the Hollywood-style trailer, we will use diagrams continuously to see that quantum theory exhibits behaviours that are simply not possible in classical physics.

Q3: *What about infinite-dimensional Hilbert spaces?*

A: The lesson of quantum computing and quantum information processing is that we can access many new, revolutionary features of quantum theory by restricting to finite (and often just two!) dimensions. In fact, the long-held belief that 'real physics' only happens in the infinite-dimensional realm, with all of its associated difficulties, could have contributed to the blindness to new quantum features that were not discovered until the past couple of decades. Of course, we are not claiming that infinite dimensions should therefore be ignored, but it does present a unique set of difficulties for quantum pictorialism. Most notably, the main diagrammatic workhorses in this book (cups, caps, and spiders) simply don't exist as bounded operators between infinite-dimensional Hilbert spaces. So, for many years, we thought one had to choose between the power (and complexity) of infinite dimensions or the elegance of diagrammatic reasoning with cups, caps, and spiders. However, certain constructions in this book that seem to rely on caps and cups can be done in ways to avoid them altogether (Coecke and Heunen, 2011), and recent results of Gogioso and Genovese (2016) suggest it is possible for us to have our caps and eat them too! Namely, they showed using techniques from non-standard analysis that, while cups, caps, and spiders don't actually exist in infinite dimensions, reasoning with them is still sound, as long as they don't appear in the final answer that comes out.

While this new way of working with infinite-dimensional systems is still in its infancy, it shows promise for the advent of a true infinite-dimensional quantum pictorialism.

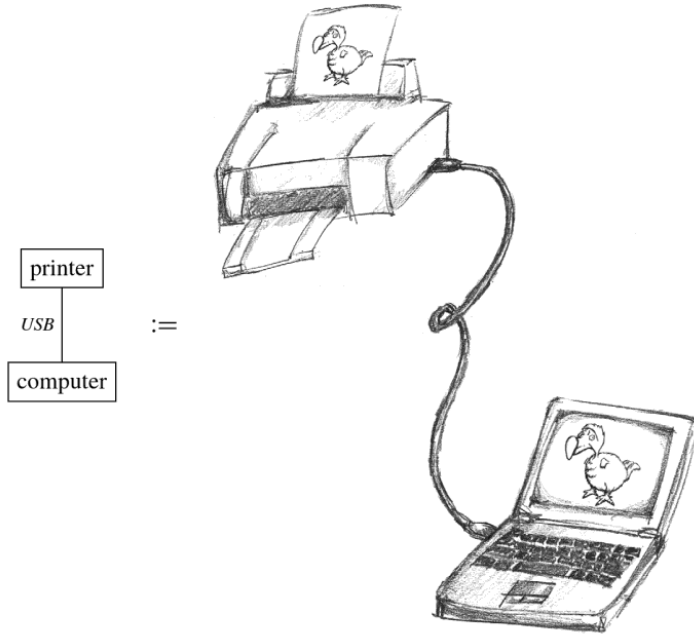
Q4: *What about Schrödinger's equation?*

A: You will notice that we are careful to use the term 'quantum theory', as opposed to 'quantum mechanics', which we take to mean the core of quantum mechanics that ignores things like positions, momenta, and continuous time-evolution. For our purposes, it suffices to just consider the overall change of systems between some time t_1 and some time t_2 , without expounding on the details of what exactly happens in between these times. As in the case of finite dimensions, the huge advances in quantum information/computation have shown us that, even working at this level, we can access many fascinating features of the theory. That being said, there has recently been some exciting new research to accommodate dynamics in quantum pictorialism by Gogioso (2015b,c), and it seems likely that many of the features we discuss in this book (e.g. strong complementarity) play a major role.

Q5: *Has quantum pictorialism ever produced anything new?*

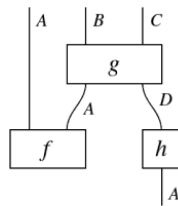
A: What people posing this question usually mean is: Has quantum pictorialism helped solve problems that were already out there and that one couldn't solve with other existing methods? The answer is yes, and some examples of that are Duncan and Perdrix (2010), Coecke et al. (2011b), Horsman (2011), and Boixo and Heunen (2012). However, in science, coming up with new, interesting questions is sometimes more important than answering old ones. Issues such as completeness of calculi is a question that to our knowledge has never been asked before in physics, and quantum pictorialism has meanwhile produced a string of results in that area (Backens, 2014a,b; Schröder de Witt and Zamdzhiev, 2014; Hadzihasanovic, 2015). It would be difficult to see how any answer could arise when sticking to the traditional Hilbert space formalism. Another new question is whether we can automate reasoning about physics, for which the Quantomatic software discussed in Chapter 14 has been produced (Kissinger and Zamdzhiev, 2015). Finally, there is the communality of structure and use of quantum pictorialism methods elsewhere, like in developing a compositional distributional model of natural language meaning (Clark et al., 2014; Coecke, 2016), which has actually outperformed other existing methods when applied to empirical data (Grefenstette and Sadrzadeh, 2011; Kartsaklis and Sadrzadeh, 2013).

In some cases, wires in the diagram may even correspond to actual physical wires, for example:

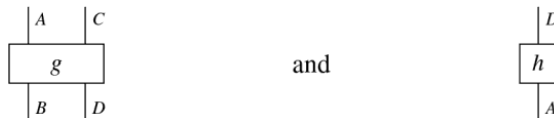


Though, of course, this needs not always be the case. Wires can also represent e.g. ‘aligning apertures’ on lab equipment, or shipping something by boat. The important thing is that wires represent the flow of data (or more general ‘stuff’) from one process to another.

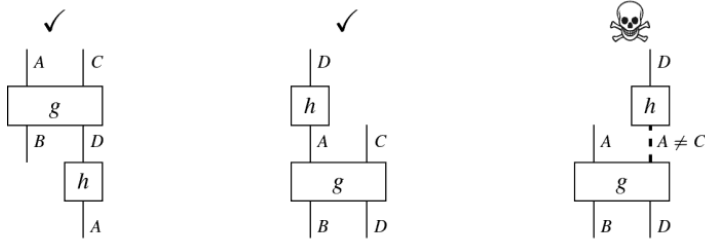
As we have just seen, we can *wire together* simple processes to make more complicated processes, which are described by *diagrams*:



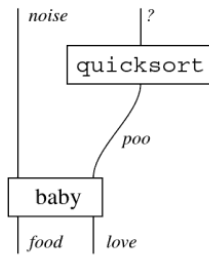
In such a diagram outputs can be wired to inputs only if their types match. For example, the following two processes:



can be connected in some ways, but not in others, depending on the types of their wires:



This restriction on which wirings are allowed is an essential part of the language of diagrams, in that it tells us when it makes sense to apply a process to a certain system and prevents occurrences like this:

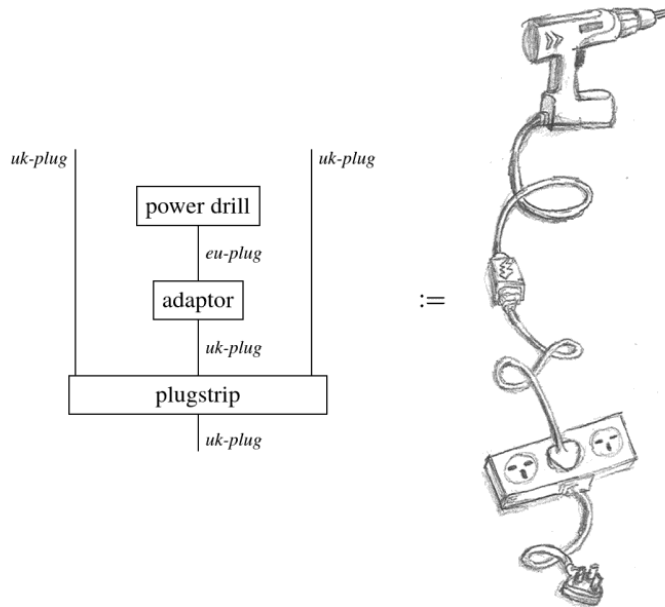


which probably wouldn't be very good for your computer! Much like *data types* in computer science, the types on wires tell us what sort of data (or stuff) the process expects as input and what it produces as output. For example, a calculator program expects numbers as inputs and produces numbers as outputs. Thus, it can't make sense of 'Dave' as input, nor will it ever produce, say, 'carrots' as an output.

Another useful example is that of electrical appliances. Suppose we consider processes to be appliances that have plugs as inputs (i.e. where electricity is 'input') and sockets as outputs:



Then, system-types are just the shape of the plug (UK, European, etc.). Clearly we can't connect a plug to a socket of the wrong shape, so the type information on wires gives us precisely the information we need to determine which wirings are possible. For example, one possible wiring is:



Though this is a toy example, it is often very useful to draw diagrams whose boxes refer to devices that actually exist in the real world (e.g. in a lab). We will refer to such diagrams as *operational*.

3.1.2 Process Theories

Usually one is not interested in all possible processes, but rather in a certain class of related processes. For example, practitioners of a particular scientific discipline will typically only study a particular class of processes: physical processes, chemical processes, biological processes, computational processes, mathematical processes, etc. For that reason, we organise processes into *process theories*. Intuitively, a process theory tells us how to interpret wires and boxes in a diagram (cf. (i) and (ii) below) and what it means to form diagrams, that is, what it means to wire boxes together (cf. (iii) below).

Definition 3.1 A *process theory* consists of:

- (i) a collection T of *system-types* represented by wires,
- (ii) a collection P of *processes* represented by boxes, where for each process in P the input types and output types are taken from T , and
- (iii) a means of ‘wiring processes together’, that is, an operation that interprets a diagram of processes in P as a process in P .

In particular, (iii) guarantees that

process theories are ‘closed under wiring processes together’,

since it is this operation that tells us what ‘wiring processes together’ means. In some cases this operation consists of literally plugging things together with physical wires, as in the example of the power drill. In other cases this will require some more work, and sometimes there is more than one obvious choice available. We shall see in Section 3.2 that in traditional mathematical practice one typically breaks down ‘wiring processes together’ in two sub-operations: parallel composition and sequential composition of processes.

Example 3.2 Some process theories we will encounter are:

- **functions** (types = sets)
- **relations** (types = sets, again)
- **linear maps** (types = vector spaces, or Hilbert spaces)
- **classical processes** (types = classical systems)
- **quantum processes** (types = quantum and classical systems)

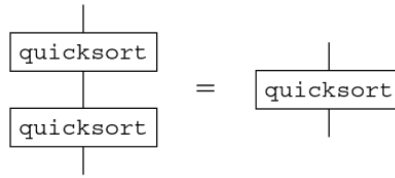
We will use the first two mainly in various examples and exercises, whereas the remaining three will play a major role in this book.

Remark 3.3 Note how we refer to a particular process theory by saying what the ‘processes’ are, leaving the types implicit. This tends to be a good practice, because the processes are the important part. For instance, we’ll see that the process theory of functions is quite different from that of relations, so it will not do to just call both ‘sets’.

Since a process theory tells us how to interpret diagrams as processes, it crucially tells us when two diagrams represent the same process. For example, suppose we define a simple process theory for **computer programs**, where the types are data-types (e.g. integers, booleans, lists) and the processes are computer programs. Then, consider a short program, which takes a list as input and sorts it. It might be defined this way (don’t worry if you can’t read the code, neither can half of the authors):

$$\begin{array}{c} | \\ \boxed{\text{quicksort}} \\ | \end{array} := \begin{cases} \text{qs } [] = [] \\ \text{qs } (x :: xs) = \\ \quad \text{qs } [y \mid y <- xs; y < x] ++ [x] ++ \\ \quad \text{qs } [y \mid y <- xs; y \geq x] \end{cases}$$

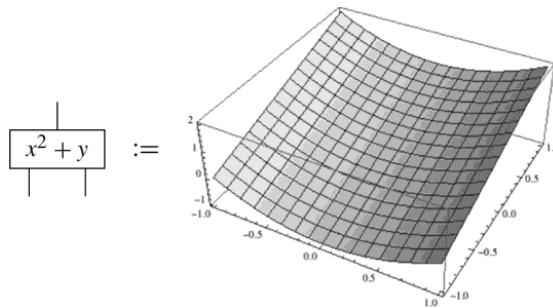
Wiring together programs means sending the output of one program to the input of another program. Taking two programs to be equal if they behave the same (disregarding some details like execution time, etc.), our process theory yields equations like this one:



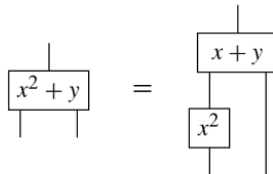
i.e. sorting a list twice has the same effect as sorting it once.

The reason we call a process theory a *theory* is that it comes with lots of such equations, and these equations are precisely what allows us to draw conclusions about the processes we are studying.

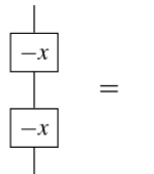
To take another example, the function we defined at the beginning of Section 3.1.1 lives in the process theory of **functions**. Types are sets and processes are functions between sets. We consider two functions equal if they behave the same; i.e. they have the same graph:



(Note how two input wires means a function of two variables, hence the three-dimensional plot.) Since they have the same graph, we have:



as well as:



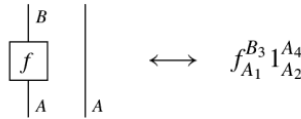
Exercise 3.4 Cook up some of your own process theories. For each one, answer the following questions:

1. What are the system-types?
2. What are the processes?

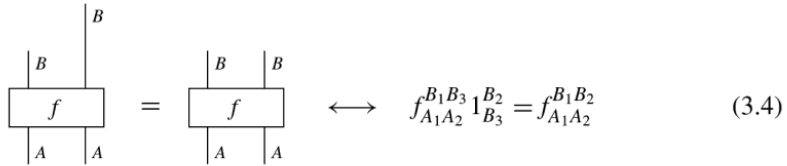
We will also consider a single wire of type A as a ‘special’ box:



with a corresponding box name $1_{A_1}^{A_2}$. Hence we can write, e.g.:



In almost every respect, these special boxes are just boxes like any other and may be interpreted as the process that ‘does nothing’. However, when we connect them to the input or output of another box, they vanish:



The following definition summarises all of above.

Definition 3.8 A *diagram formula* is a sequence of box names such that all wire names are unique, except for matched pairs of upper and lower names. Two diagram formulas are equal if and only if one can be turned into the other by:

- (a) changing the order in which box names are written,
- (b) adding or removing identity boxes, as in (3.4), or
- (c) changing the name of a repeated wire name.

With this concept in hand, we can now provide a definition of a diagram that makes direct reference to a formula-like counterpart.

Definition 3.9 A *diagram* is a pictorial representation of a diagram formula where box names are depicted as boxes (or various other shapes), wire names are depicted as input and output wires of the boxes, and repeated wire names tell us which outputs are connected to which inputs.

Exercise 3.10 Draw the diagrams of the following diagram formulas:

$$f_{B_1 C_2}^{C_4} g_{C_4}^{D_3} \quad f_{A_1}^{A_1} \quad g_{B_1}^{A_1} f_{A_1}^{B_1} \quad 1_{A_1}^{A_6} 1_{A_2}^{A_5} 1_{A_3}^{A_4}$$

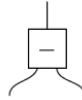
Use the convention that inputs and outputs are numbered from left to right.

The above construction shows that all of our work with diagrams could be translated, without ambiguity, into work on diagram formulas. We will mostly stick to diagrams since they are easier to visualise, are easier to work with, and do away with the bureaucratic overhead of extra subscripts. However, we will occasionally use diagram formulas as a handy tool for computing the process of a given diagram, as in Sections 3.3.3 and 5.2.4.

3.1.4 Process Equations

In Example 3.6 above, we wrote down many equations involving diagrams on the left-hand side (LHS) and the right-hand side (RHS). These *diagram equations* always hold, regardless of how we interpret the boxes and wires. That is, they are true in any process theory. On the other hand, within a particular process theory it may be possible to represent the same *process* using two distinct diagrams. We have already seen a couple of examples of this in Section 3.1.2.

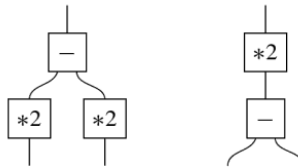
Returning to the process theory **functions**, suppose we define two processes: ‘minus’, which takes two inputs m, n and subtracts them, outputting $m - n$:



and ‘times 2’, which takes a single input and multiplies it by 2:



where all of the wires have type \mathbb{R} . Now, consider the following two diagrams:



As *diagrams*, these two are not equal. However, if we look at the *processes* they represent, we see that the process on the left multiplies both inputs by 2, then subtracts one from the other. The one on the right first subtracts the inputs, then multiplies the result by 2. Even though they have different diagrams, both processes compute the same function, namely:

$$2m - 2n = 2(m - n)$$

Thus, these distinct diagrams represent the same process when interpreted in the process theory of **functions**; i.e. they are *equal as processes*, which we write simply as:

$$(3.5)$$

This is called a *process equation*. A diagram equation is a (trivial) special case of a process equation, since equal diagrams will always be interpreted as equal processes.

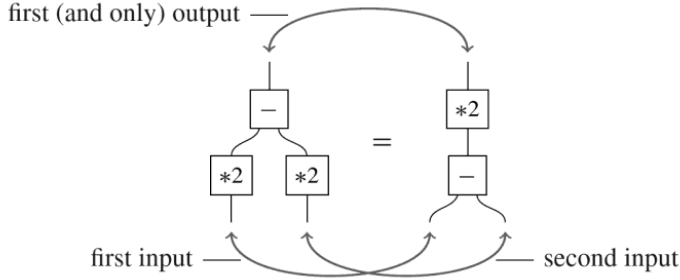
Remark* 3.11 There is a special class of theories where diagram equations and process equations coincide. That is, two processes are equal if and only if they have the same diagram. These are called *free process theories*, in analogy with the use of the term ‘free’ in algebra. Here, ‘free’ means ‘no extra equations’. In general, imposing extra equations between processes puts a constraint on which interpretations of boxes are allowed (i.e. only those satisfying the extra equations). Free process theories have the property that any interpretation of their boxes as processes (in some other process theory) extends to a consistent interpretation of diagrams.

Exercise 3.12 Give the (diagram) equations that express the algebraic properties of associativity, unitality, and commutativity of a two-input, one-output process. Can you do the same for distributivity of a pair of two-input processes (e.g. ‘plus’ and ‘times’)? If not, what’s the problem?

The notion of a diagram equation may seem completely obvious (because it is!), but it is important to note that such an equation gives a bit more information than you might first think. To see why, note that equation (3.5) is true, but the following equation is **false**:

$$(3.6)$$

The LHS will map the inputs m, n to $2m - 2n$, whereas the RHS will map those same inputs to $2(n - m) \neq 2m - 2n$. The only difference between this equation and the true equation (3.5) is that the inputs of the RHS are flipped. Even though we do not write down the names of wires in the diagrams (as we do for the diagram formulas of Definition 3.8), these inputs and outputs do have distinct identities. Furthermore, for a diagram equation to be well formed, both sides must have the *same* inputs and outputs, which suggests that there is a correspondence between them:



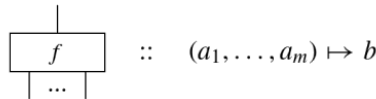
In terms of diagram formulas, this would be reflected by the fact that we call e.g. the first input A_1 and the second input A_2 on both sides of the equation. In a diagram, we show which inputs and outputs are in correspondence by their positions on the page. Bearing this rule in mind, there is clearly a difference between the correct equation (3.5) and the erroneous (3.6).

Equations between processes with only one ‘output’ should already be familiar from algebra. When we write an equation like:

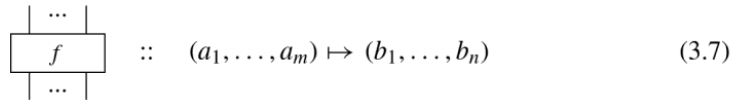
$$2m - 2n = 2(m - n)$$

we distinguish the inputs m, n of the formulas on the LHS and RHS by giving them names: namely ‘ m ’ and ‘ n ’. In algebra, there is always exactly one ‘output’ of a formula, namely the value computed when all the variables have been substituted by numbers, so we don’t bother to give it a name. On the other hand, diagrams can have *many* outputs in general, so we need to distinguish those as well.

We typically think of multivariable functions, such as the algebraic operations above, as taking one or more inputs to a single output:



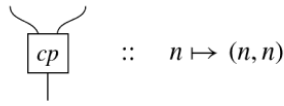
where the ‘ \mapsto ’ symbol means ‘maps to’ (cf. Appendix). However, in the process theory **functions** we can also consider functions of this form:



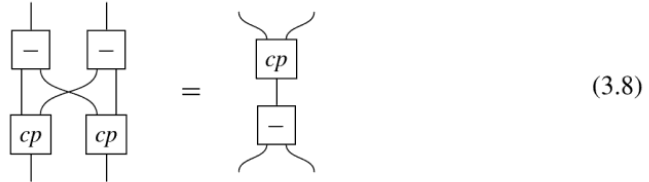
Remark* 3.13 In functional programming, one might encounter a function with many outputs in the sense of (3.7) in expressions like:

$$\text{‘let } (b_1, \dots, b_n) = f(a_1, \dots, a_m) \text{ in ...’}$$

A simple example of a two-output function is cp , which takes in a number n and sends a copy of n down both of its output wires:



The function cp satisfies the following process equation with minus:



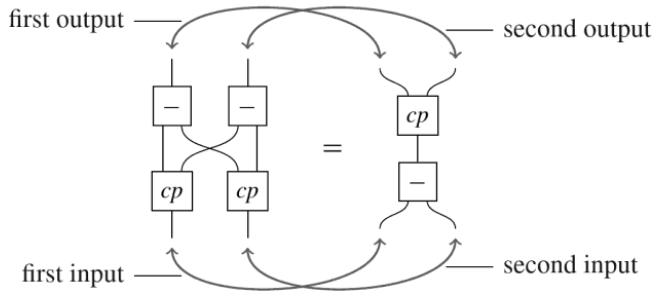
The process on the LHS takes in two inputs m and n , copies each of them, and sends one copy of each to two separate subtraction operations. So, the output is two copies of $m - n$, one on each wire:

$$(m, n) \mapsto (m, m, n, n) \mapsto (m, n, m, n) \mapsto (m - n, m - n)$$

The process on the RHS takes the inputs m and n , subtracts them, and copies the result. Again, this yields two copies of $m - n$:

$$(m, n) \mapsto m - n \mapsto (m - n, m - n)$$

As before, implicit in equation (3.8) is a correspondence between inputs and outputs:



Exercise 3.14 Write (3.8) as an equation between diagram formulas.

Exercise* 3.15 Using cp , is it now possible to express the distributivity equation between ‘plus’ and ‘times’ mentioned in Exercise 3.12?

3.1.5 Diagram Substitution

We can obtain new process equations from old ones via *diagram substitution*. Along with simple diagram deformation, diagram substitution will be the most common style of calculation in this book.

3.2 Circuit Diagrams

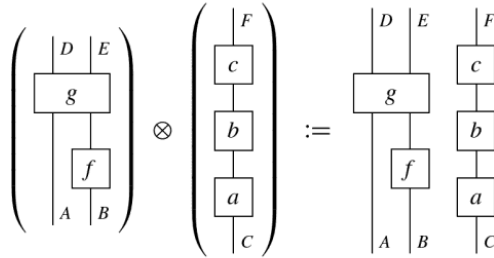
Recalling that boxes represent processes, we can define two basic *composition operations* on processes with the following interpretations:

$$\begin{aligned}
 f \otimes g &:= \text{'process } f \text{ takes place while process } g \text{ takes place' } \\
 f \circ g &:= \text{'process } f \text{ takes place after process } g \text{ takes place' }
 \end{aligned}$$

These operations allow us to define an important class of diagrams called *circuits*. Despite their importance, we will see in the next chapter that the most interesting processes are in fact those that fail to be circuits.

3.2.1 Parallel Composition

The *parallel composition* operation consists of placing a pair of diagrams side by side. We write this operation using the symbol ' \otimes ':



Any two diagrams can be composed in this manner, since placing diagrams side by side does not involve connecting anything. This reflects the intuition that both processes are happening independently of each other.

This composition operation is associative:

$$\left(\begin{array}{|c|} \hline \square \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \square \\ \hline \end{array} \right) \otimes \begin{array}{|c|} \hline \square \\ \hline \end{array} = \begin{array}{|c|} \hline \square \\ \hline \end{array} \begin{array}{|c|} \hline \square \\ \hline \end{array} \begin{array}{|c|} \hline \square \\ \hline \end{array} = \begin{array}{|c|} \hline \square \\ \hline \end{array} \otimes \left(\begin{array}{|c|} \hline \square \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \square \\ \hline \end{array} \right) \quad (3.11)$$

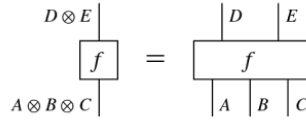
and it has a unit, the empty diagram:

$$\begin{array}{|c|} \hline \square \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \square \\ \hline \end{array} = \begin{array}{|c|} \hline \square \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \square \\ \hline \end{array} = \begin{array}{|c|} \hline \square \\ \hline \end{array} \quad (3.12)$$

Parallel composition is defined for system-types as well. That is, for types A and B , we can form a new type $A \otimes B$, called the *joint system-type*:

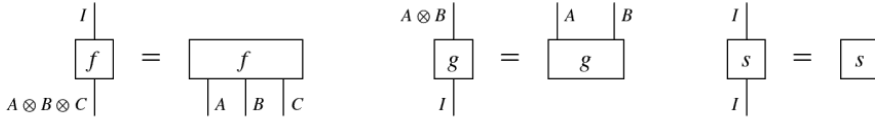
$$A \otimes B \quad \Big| \quad := \quad \Big| \quad A \quad \Big| \quad B$$

We have already made use of composition of system-types implicitly by drawing boxes that have many inputs and outputs. Formally, a box with three inputs A, B, C and two outputs D and E is the same as a box with a single input $A \otimes B \otimes C$ and a single output $D \otimes E$:



There is also a special ‘empty’ system-type, symbolically denoted I , which is used to represent ‘no inputs’, ‘no outputs’, or both.

Examples 3.18 Here are some examples of boxes with no input and/or output wires, written both in terms of ‘ \otimes ’ and I and in the usual way:

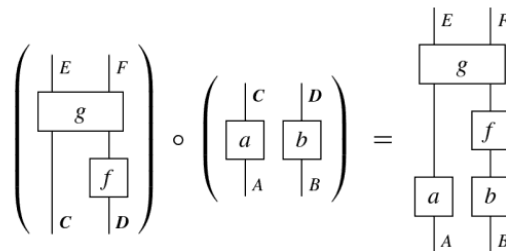


Processes with no inputs and/or outputs play a special role in this book, as we will see starting in Section 3.4.1.

In the diagrammatic notation, we will rarely use the \otimes symbol explicitly, preferring instead to express joint systems as multiple wires.

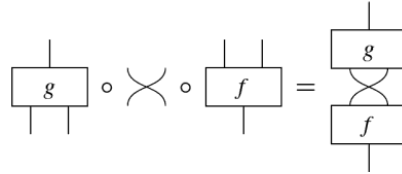
3.2.2 Sequential Composition

The *sequential composition* operation consists of connecting the outputs of a first diagram to the inputs of a second diagram. We write this operation using the symbol ‘ \circ ’:



The intuition is that the process on the right happens first, and then the process on the left happens, taking the output of the first process as its input. Clearly not any pair of diagrams can be composed in this manner: the number and type of the inputs of the left process must match the number and type of the outputs of the right process.

We assume that sequential composition will connect outputs to inputs in order. If we wish to vary this order, we can introduce *swaps*:



The LHS of this equation is well-defined of course, since the sequential composition operation is also associative:

$$\left(\begin{array}{c} | \\ \boxed{h} \\ | \end{array} \circ \begin{array}{c} | \\ \boxed{g} \\ | \end{array} \right) \circ \begin{array}{c} | \\ \boxed{f} \\ | \end{array} = \begin{array}{c} | \\ \boxed{h} \\ | \\ \boxed{g} \\ | \\ \boxed{f} \\ | \end{array} = \begin{array}{c} | \\ \boxed{h} \\ | \end{array} \circ \left(\begin{array}{c} | \\ \boxed{g} \\ | \end{array} \circ \begin{array}{c} | \\ \boxed{f} \\ | \end{array} \right) \quad (3.13)$$

and it also has a unit. This time, it's a plain wire of appropriate type:

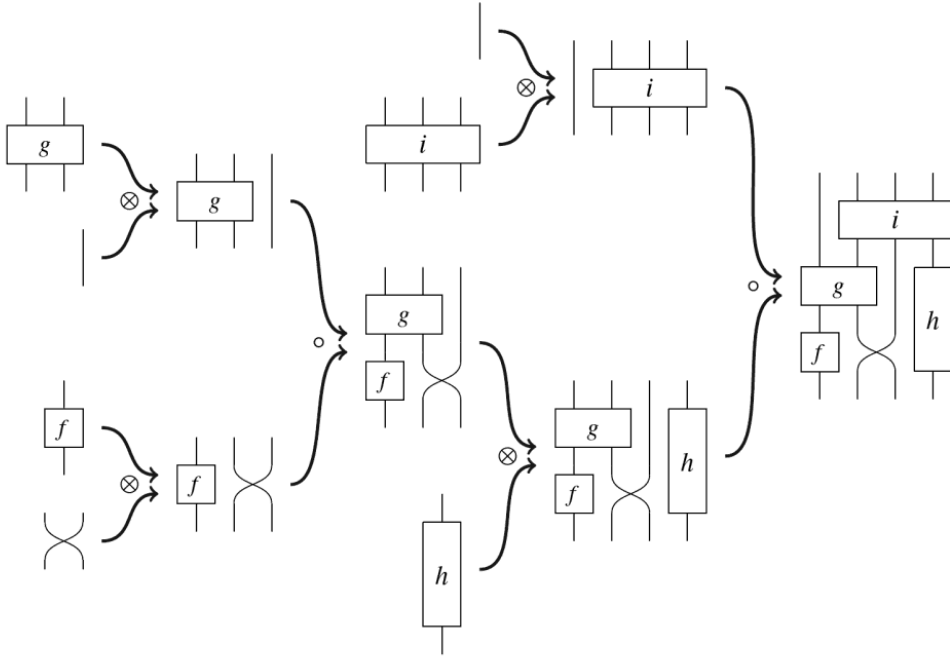
$$\begin{array}{c} | \\ \circ \\ | \end{array} \begin{array}{c} | \\ \boxed{f} \\ | \\ A \end{array} = \begin{array}{c} | \\ \boxed{f} \\ | \\ A \end{array} \begin{array}{c} | \\ \circ \\ | \\ A \end{array} = \begin{array}{c} | \\ \boxed{f} \\ | \\ A \end{array} \quad (3.14)$$

We already encountered these plain wires or *identities* in Section 3.1.3, where we mentioned that as a process, we can think of them as ‘doing nothing’ to the input. For a system of type A we will denote the identity on A symbolically as 1_A . The empty diagram is also an identity, but on the system-type I , so we denote it as 1_I .

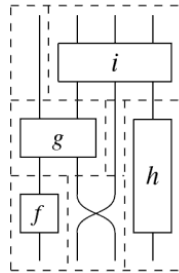
3.2.3 Two Equivalent Definitions of Circuits

Definition 3.19 A diagram is a *circuit* if it can be constructed by composing boxes, including identities and swaps, by means of \otimes and \circ .

Every diagram that we have seen in this chapter is in fact a circuit. Here is an example of the assembly of such a circuit:



We can still recognise this assembly structure in the resulting diagram:



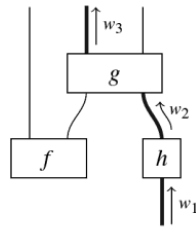
(3.15)

Note however that such a decomposition of a diagram does not uniquely determine the assembly process (e.g. associativity of \otimes and \circ allows for two manners of composing three boxes), and the same diagram may even allow for several distinct decompositions. We will see in the next section how this feature makes a non-diagrammatic treatment of circuits in terms of \otimes and \circ especially unwieldy.

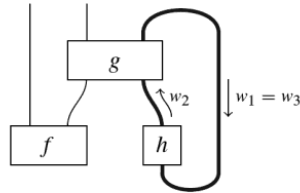
Not all diagrams are circuits. To understand which ones are, we provide an equivalent characterisation that doesn't refer to the manner that one can build these diagrams but instead to a property that has to be satisfied.

Definition 3.20 A *directed path* of wires is a list of wires (w_1, w_2, \dots, w_n) in a diagram such that for all $i < n$, the wire w_i is an input to some box for which the wire w_{i+1} is an output. A *directed cycle* is a directed path that starts and ends at the same box.

An example of a directed path is shown in bold here:



and an example of a directed cycle is:

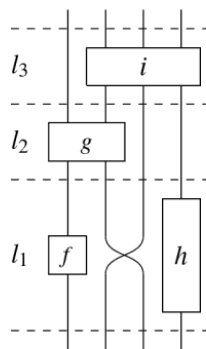


Remark* 3.21 In an area of mathematics called graph theory, graphs without directed cycles are called *directed acyclic*.

Theorem 3.22 The following are equivalent:

- a diagram is a circuit, and
- it contains no directed cycles.

Proof There exists a (non-unique) way to express any diagram with no directed cycles in terms of \otimes , \circ , and swap. Since the diagram contains no directed cycles, we divide the diagram into ‘layers’ l_1, \dots, l_n , where the outputs of each box (including identities and swaps) in layer l_i connect only to boxes in layer l_{i+1} . For example, one way to divide diagram (3.15) into layers is:



In each layer, \otimes combines boxes into one, and \circ combines the layers together, so we indeed have a circuit. Conversely, any diagram built up using just \otimes and \circ can be decomposed into such layers, which implies the absence of directed cycles. \square

Evidently, in the absence of any additional equations on the \otimes - and \circ -symbols these are not equal. Now we compute their corresponding diagrams. For the first expression we obtain:

$$\left(\begin{array}{|c|} \hline \boxed{g_1} \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \boxed{g_2} \\ \hline \end{array} \right) \circ \left(\begin{array}{|c|} \hline \boxed{f_1} \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \boxed{f_2} \\ \hline \end{array} \right) = \left(\begin{array}{|c|} \hline \boxed{g_1} \quad \boxed{g_2} \\ \hline \end{array} \right) \circ \left(\begin{array}{|c|} \hline \boxed{f_1} \quad \boxed{f_2} \\ \hline \end{array} \right) = \begin{array}{|c|} \hline \boxed{g_1} \quad \boxed{g_2} \\ \hline \boxed{f_1} \quad \boxed{f_2} \\ \hline \end{array}$$

and for the second we obtain:

$$\left(\begin{array}{|c|} \hline \boxed{g_1} \\ \hline \end{array} \circ \begin{array}{|c|} \hline \boxed{f_1} \\ \hline \end{array} \right) \otimes \left(\begin{array}{|c|} \hline \boxed{g_2} \\ \hline \end{array} \circ \begin{array}{|c|} \hline \boxed{f_2} \\ \hline \end{array} \right) = \left(\begin{array}{|c|} \hline \boxed{g_1} \\ \hline \boxed{f_1} \\ \hline \end{array} \right) \otimes \left(\begin{array}{|c|} \hline \boxed{g_2} \\ \hline \boxed{f_2} \\ \hline \end{array} \right) = \begin{array}{|c|} \hline \boxed{g_1} \quad \boxed{g_2} \\ \hline \boxed{f_1} \quad \boxed{f_2} \\ \hline \end{array}$$

So we get the same diagram twice!

Since the \otimes - and \circ -symbols are supposed to be an algebraic syntax for diagrams, we need to impose a new equation:

$$(g_1 \otimes g_2) \circ (f_1 \otimes f_2) = (g_1 \circ f_1) \otimes (g_2 \circ f_2) \tag{3.16}$$

while in the diagrammatic language this is nothing but a tautology! So what is the actual content of (3.16)? It states that the composite process:

process g_1 takes place while process g_2 takes place,
after
process f_1 takes place while process f_2 takes place,

is the same as the composite process:

process g_1 takes place after process f_1 takes place,
while
process g_2 takes place after process f_2 takes place.

Evidently this is true, and it is something we get for free in the diagrammatic language, but in algebraic language we need to explicitly state this.

Given the equations we have stated thus far for the \otimes - and \circ -symbols, we can derive many new ones. For example:

$$\begin{array}{|c|} \hline \boxed{g} \\ \hline \end{array} \circ \begin{array}{|c|} \hline \boxed{f} \\ \hline \end{array} = \begin{array}{|c|} \hline \boxed{g} \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \boxed{f} \\ \hline \end{array}$$

can be derived by combining (3.12), (3.14), and (3.16):

$$\begin{array}{|c|} \hline \boxed{g} \\ \hline \end{array} \circ \begin{array}{|c|} \hline \boxed{f} \\ \hline \end{array} = \left(\begin{array}{|c|} \hline \boxed{g} \\ \hline \end{array} \otimes 1_I \right) \circ \left(1_I \otimes \begin{array}{|c|} \hline \boxed{f} \\ \hline \end{array} \right) = \left(\begin{array}{|c|} \hline \boxed{g} \\ \hline \end{array} \circ 1_I \right) \otimes \left(1_I \circ \begin{array}{|c|} \hline \boxed{f} \\ \hline \end{array} \right) = \begin{array}{|c|} \hline \boxed{g} \\ \hline \end{array} \otimes \begin{array}{|c|} \hline \boxed{f} \\ \hline \end{array}$$

while in the diagrammatic language it is again a tautology. Another example of an extra equation for the \otimes - and \circ -symbols involves a crossing:

$$(3.17)$$

One can easily imagine how many more non-trivial equations are derivable when combining (3.11), (3.12), (3.14), (3.16), and (3.17), many of which would look not at all obvious without drawing the diagram.

Exercise* 3.26 Assume the following equation:

$$\sigma_{A \otimes B, C} = (\sigma_{A, C} \otimes 1_B) \circ (1_A \otimes \sigma_{B, C}) \quad \text{where} \quad \sigma_{A, B} := \begin{array}{c} \text{B} \quad \text{A} \\ \diagdown \quad \diagup \\ \text{A} \quad \text{B} \end{array}$$

Prove the first and the last diagram equation from Example 3.6 as algebraic equations using just the above equation and the other algebraic equations introduced in this section.

So why do things become so complicated? In simple terms, one is trying to squeeze something that wants to live in two dimensions (a diagram) into one dimension (a ‘linear’ algebraic notation). When we do this, the horizontal and the vertical space on the piece of paper we used to draw our diagram suddenly coincide, and we need a bunch of extra syntax (e.g. brackets), to disambiguate the parallel and sequential compositions in their new, ‘compressed’ world. We then need to subject this extra syntax to a bunch of extra rules to enable the kinds of deformations we were doing quite naturally with diagrams before. The punchline:

Diagrams rule!

There is a lot more to say about the precise connection between algebraic equations and diagrammatic reasoning, particularly in the context of an area called (*monoidal*) *category theory*. Although the latter is not crucial to understanding this book, we refer the interested reader to the advanced material in Section 3.6.2.

3.3 Functions and Relations as Processes

We now introduce two simple process theories: the theory of **functions** and the theory of **relations**. Besides it being useful to have some concrete examples of process theories in hand in order to understand some of the concepts to come, these examples establish two important insights:

- that traditional mathematical structures (sets, in this case) can form the types of a process theory, and

- that, even for some fixed system-types, the choice of processes is in fact far more important in determining the character of a process theory. In particular, while the process theories **functions** and **relations** are both based on sets, we shall see in this and the next chapter that their properties as process theories couldn't be farther apart.

For these reasons, these example process theories and, maybe somewhat surprisingly, **relations** in particular, will prove a useful stepping stone towards the process theories of **linear maps** and **quantum maps**.

3.3.1 Sets

To define a process theory, we need to first say what the system-types are, then what the processes are. For both **functions** and **relations**, the system-types are just sets. We will encounter some familiar sets, like the natural numbers \mathbb{N} , the real numbers \mathbb{R} , the complex numbers \mathbb{C} (whose properties we review in Section* 5.3.1), and:

$$\mathbb{B} := \{0, 1\}$$

which is called the set of *booleans* or *bit values*. Joint system-types are formed by taking the *Cartesian product* of sets, that is:

$$A \otimes B := A \times B = \{(a, b) \mid a \in A, b \in B\}$$

The definition is similar for three or more sets, in which case we replace pairs of elements by *tuples*:

$$A_1 \times \cdots \times A_n := \{(a_1, \dots, a_n) \mid a_i \in A_i\}$$

If we ignore brackets on tuples of elements, e.g. letting:

$$((a, b), c) = (a, b, c) = (a, (b, c))$$

parallel composition of systems is associative:

$$A \times (B \times C) = (A \times B) \times C$$

as required.

Example 3.27 The set of *bitstrings* of length n can be expressed as an n -fold Cartesian product:

$$\underbrace{\mathbb{B} \times \cdots \times \mathbb{B}}_n$$

We can think of this as the set of natural numbers ranging from 0 to $2^n - 1$ by considering the bit strings as a binary representations of these numbers; e.g. for $n = 4$ we represent 0 as $(0, 0, 0, 0)$, 1 as $(0, 0, 0, 1)$, 2 as $(0, 0, 1, 0)$, ..., up to 15, which we represent as $(1, 1, 1, 1)$. This is a trick often used in computer science, and it is also important in this book.

The trivial type I is defined to be the set $\{*\}$ that contains just a single element, here called ‘*’. If we always drop * from tuples, e.g. letting:

$$(a, *) = a = (*, a)$$

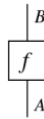
then $\{*\}$ becomes the unit for parallel composition of wires:

$$A \times \{*\} = A = \{*\} \times A$$

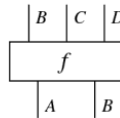
Remark* 3.28 Strictly speaking, the sets $(A \times B) \times C$ and $A \times (B \times C)$, as well as the sets $A \times \{*\}$ and A , are not equal, but *isomorphic*; that is, their elements are in one-to-one correspondence. In fact, they are not just isomorphic but isomorphic in a very strong sense, called *naturally isomorphic*, which pretty much means that for all practical purposes they can be treated as equal. The failure of strict equality is mainly due to the sort of ‘algebraic bureaucracy’ we complained about in Section 3.2.4, so it won’t have any effect on working with diagrams. Section* 3.6.2 contain more details on this somewhat delicate issue.

3.3.2 Functions

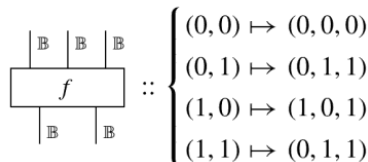
In the process theory of **functions** every process:



is a function from a set A to a set B . Since joint systems are formed using the Cartesian product, a function with many inputs and outputs, e.g.



is a function from the Cartesian product of sets $A \times B$ to the Cartesian product $B \times C \times D$. So f maps pairs $(a, b) \in A \times B$ to triples $(b', c, d) \in B \times C \times D$. More specifically, a function from $\mathbb{B} \times \mathbb{B}$ to $\mathbb{B} \times \mathbb{B} \times \mathbb{B}$ is a function from bit strings of length 2 to bit strings of length 3, for example:



Now that we know what the system-types and processes are, we need to say what ‘wiring processes together’ means (cf. Definition 3.1). We will do so by specifying what

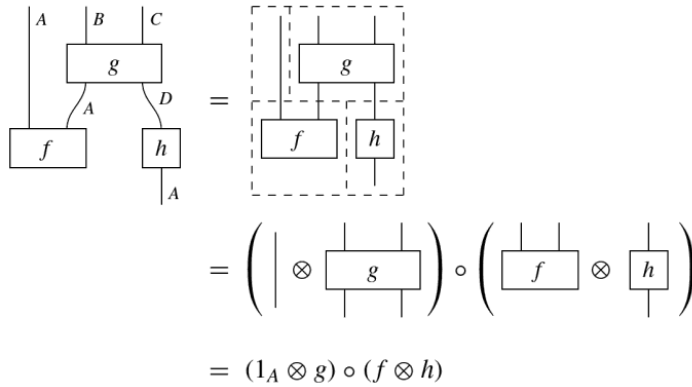
parallel and sequential composition for functions means. Sequential composition is the usual composition of functions:

$$(g \circ f)(a) := g(f(a)) \tag{3.18}$$

Parallel composition is defined as applying each function to its corresponding element of a pair:

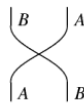
$$(f \otimes g)(a, b) := (f(a), g(b)) \tag{3.19}$$

This tells us everything we need to know to compute the function represented by a circuit diagram, since we can decompose the diagram across \otimes and \circ :



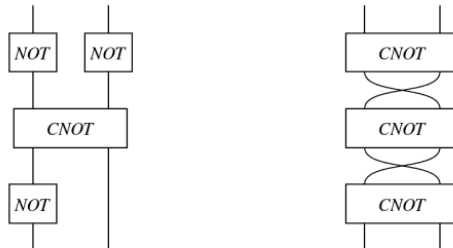
then applying equations (3.18) and (3.19). There might be more than one way to decompose a diagram, but the actual function computed will always be the same. Note that this will work for any diagram where the wires do not cross. To handle crossings, we need to define a ‘swap’ function.

Exercise 3.29 Which function represents the swap:



where A and B can be any set?

Exercise 3.30 Compute the functions:



Step 1: Write P as a diagram formula:

$$P_{B_1 A_1}^{A_2 B_2 C_1} = R_{B_1}^{A_2 A_3} S_{A_3 D_1}^{B_2 C_1} T_{A_1}^{D_1}$$

Step 2: Change wire names to elements of the appropriate set. For example, write B_2 as an element $b_2 \in B$:

$$P_{b_1 a_1}^{a_2 b_2 c_1} \iff R_{b_1}^{a_2 a_3} S_{a_3 d_1}^{b_2 c_1} T_{a_1}^{d_1}$$

Step 3: Change box names to actual relations that map the element(s) in the subscript to the element(s) in the superscript, and adjoin ‘ $\exists x$ ’ for every repeated element x :

$$P :: (b_1, a_1) \mapsto (a_2, b_2, c_1) \iff \exists a_3 \exists d_1 \left(\begin{array}{l} R :: b_1 \mapsto (a_2, a_3) \text{ and} \\ S :: (a_3, d_1) \mapsto (b_2, c_1) \text{ and} \\ T :: a_1 \mapsto d_1 \end{array} \right)$$

In the absence of inputs and/or outputs, one must use ‘*’, i.e. the unique element in the set $I = \{*\}$.

Step 4: The expression in the RHS of ‘ \iff ’ now uniquely characterises all related tuples in the LHS, and hence can now be used to compute P .

For the particular cases of sequential and parallel composition of relations, we recover (3.21) and (3.22), respectively. To see this, just draw the diagrams of $f \otimes g$ and $g \circ f$, apply steps 1–3, and see what comes out.

Exercise 3.31 Suppose A, B, C , and D are the following sets:

$$A = \{a_1, a_2, a_3\}$$

$$B = \mathbb{B}$$

$$C = \{\mathbf{red}, \mathbf{green}\}$$

$$D = \mathbb{N}$$

Compute P as defined by (3.23), for R, S, T defined as follows:

$$R :: \begin{cases} 1 \mapsto (a_1, a_1) \\ 1 \mapsto (a_1, a_2) \end{cases} \quad S :: \begin{cases} (a_1, 5) \mapsto (0, \mathbf{red}) \\ (a_1, 5) \mapsto (1, \mathbf{red}) \\ (a_2, 6) \mapsto (1, \mathbf{green}) \end{cases} \quad T :: \begin{cases} a_1 \mapsto 200 \\ a_3 \mapsto 5 \end{cases}$$

and for R, S, T defined as follows:

$$R :: \begin{cases} 0 \mapsto A \times \{a_2, a_3\} \\ 1 \mapsto A \times \{a_2, a_3\} \end{cases} \quad S :: \begin{cases} (a_1, 0) \mapsto \mathbb{B} \times \{\mathbf{red}, \mathbf{green}\} \\ (a_1, 1) \mapsto \mathbb{B} \times \{\mathbf{red}, \mathbf{green}\} \\ (a_1, 2) \mapsto \mathbb{B} \times \{\mathbf{red}, \mathbf{green}\} \\ \vdots \end{cases} \quad T :: \begin{cases} a_1 \mapsto \mathbb{N} \\ a_2 \mapsto \mathbb{N} \\ a_3 \mapsto \mathbb{N} \end{cases}$$

Remark* 3.32 Due to the close resemblance between the process theories **relations** and **linear maps**, and in particular, the fact that they both admit a *matrix calculus* (see Chapter 5), this ‘trick’ for computing diagrams of relations applies also for computing diagrams of linear maps, as we shall see in Section 5.2.4, and hence will also be relevant for **quantum processes**. In fact, if we treat relations as matrices with boolean entries, the computation is essentially identical.

3.3.4 Functions versus Relations

When we restrict the definitions (3.21) and (3.22) to functions, we obtain (3.18) and (3.19), respectively. In both cases, we took advantage of the fact that functions relate each input element to a unique output to obtain a simpler form. For example, in the case of parallel composition of functions $f : A \rightarrow C$ and $g : B \rightarrow D$, for each $a \in A$ and each $b \in B$ there is a unique $c \in C$ and $d \in D$ such that $f :: a \mapsto c$ and $g :: b \mapsto d$, and hence, building $f \otimes g$ amounts to merely pairing these two elements as we did in (3.19). On the other hand, for relations, $R \otimes S :: (a, b) \mapsto (c, d)$ whenever there exist both a $c \in C$ and a $d \in D$ such that $R :: a \mapsto c$ and $S :: b \mapsto d$, and there could of course be more than one of such pair (c, d) .

Now, since:

1. both **functions** and **relations** have sets as system-types, so in particular, the system-types of **relations** include those of **functions**;
2. functions are a special case of relations, and hence the processes of **relations** include the processes of **functions**; and
3. sequential composition \circ of **functions** is a special case of sequential composition of **relations**, and parallel composition \otimes of **functions** is a special case of parallel composition of **relations**,

we say that the process theory of **functions** is a *subtheory* of **relations**. We can write this as follows:

$$\mathbf{functions} \subseteq \mathbf{relations}$$

Of course, there are many more relations than there are functions, and throughout this book we will encounter many important relations that aren’t functions. In fact, these extra relations are precisely what make the process theory of **relations** behave much more like **linear maps** than **functions**.

3.4 Special Processes

In this section, we single out several types of process that will play a central role throughout this book.

3.4.1 States, Effects, and Numbers

So far, we have been handling boxes without any inputs and/or without any outputs just like any other boxes. However, these particular boxes have a very special status when we interpret them as processes.

- *States* are processes without any inputs. In operational terms they are ‘preparation procedures’. We represent them as follows:



Whereas a system A could be in any number of possible different states, a preparation procedure will produce a system of type A in a particular state ψ taken from these possibilities. The fact that there is no wire in means we don’t know (or care) where this system came from, just that we have it now and it is in a particular state. For example, if we say ‘here is a bit initiated in the 0 state’, then we don’t really care about the previous history of that bit as long as we have the guarantee that it is currently in state 0 and that it is available to perform some further computations. Or, in the case of saying ‘here is a fresh raw potato’, we don’t really care whose farm it came from or which animal’s manure was used, as long as we know that we have a fresh raw potato that we can process further into fries.

- *Effects* are processes without any outputs. We have borrowed this terminology from quantum theory, where effects play a key role. From now on we represent them as follows:



The notion of effect is dual to the notion of state, in that one starts with a system and has nothing left afterwards – or, we simply don’t care what we have afterwards. The simplest example of an effect consists of *discarding* a system, where the system is destroyed (or simply disregarded). Less trivial effects refer to things that may or may not happen, depending on the actual state of a system. For example, ‘the bomb exploded’, ‘the photon was absorbed’, ‘all the food was eaten’, or ‘the dodo became extinct’ could all be considered effects, since afterwards there is no bomb, no photon, no food, and no dodo.

Operationally speaking, effects are used to model ‘tests’. When we say an effect ‘happens’, it means we tested whether a system satisfies a certain property and have obtained ‘yes’ as the answer, after which we discard that system, or – as is often the case – the system is destroyed in the verification process. Altogether, a test consists of:

1. a question about a system,
2. a procedure that enables one to verify this question, and
3. the event of obtaining ‘yes’ as the answer; i.e. the effect ‘happened’.

So, there is more to the notion of a test than just answering a question. For example, when making a drawing, one could ask the question: ‘Is this a red pen?’ A corresponding test would then consist of taking a piece of paper, then writing with the pen on it, and observing that the colour is indeed red. Of course, in this case one would (usually) not destroy the pen. Genuine destruction takes place when one asks the question: ‘Is this a working match?’ A corresponding test would then consist of trying to light the match and succeeding, after which we end up with no match. The act of discarding a system is also a

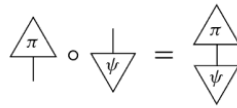
test, albeit a trivial one. After verifying some property that holds for any state of a system (e.g. ‘does this system exist?’), we discard it. This is of course a very silly test, since no information is gained, but the system is lost.

At first glance, it seems like tests are more than just ‘preparation procedures in reverse’. However, if we assume that preparation procedures can fail (as is often the case in the lab), we see that the notions really are symmetric. A preparation procedure consists of:

1. a state we would like our system to be in,
2. a procedure that enables one to put a system in that state, and
3. the event of obtaining ‘yes’; i.e. the state was successfully prepared.

Of course, unlike tests, we could just keep retrying until a preparation succeeds (or we run out of funding), so we tend to disregard part 3.

When we compose a state and an effect, a third kind of special box arises:



- *Numbers* are processes without any inputs or outputs. From now on we represent them as:



or sometimes simply as:

λ

At this point, you’re probably saying ‘Hang on! I already know what numbers are, and this doesn’t have anything to do with processes with no inputs and outputs!’ Well, the numbers you learned about in school (e.g. whole numbers, real numbers, possibly complex numbers) are all instances of this very general kind of ‘number’.

So, what are processes with no inputs or outputs? Well, they are just a set of things that can be ‘multiplied’, thanks to \otimes -composition (or equivalently in this case, \circ -composition). That is, if λ and μ are numbers, then:

$$\lambda \otimes \mu := \diamond \lambda \diamond \mu$$

is also a number. Also, we saw in Section 3.2.4 that diagrams rule, so this ‘multiplication’ is associative:

$$(\lambda \otimes \mu) \otimes \nu = \diamond \lambda \diamond \mu \diamond \nu = \lambda \otimes (\mu \otimes \nu)$$

and has a unit, given by the empty diagram:

$$\lambda \otimes 1_I = \diamond \lambda \square = \lambda$$

And finally, because of the lack of inputs and outputs numbers can move around liberally within a diagram, and even can do a little dance:

$$\begin{array}{ccccccc}
 \diamond \lambda & = & \diamond \lambda & = & \diamond \lambda & \diamond \mu & = & \diamond \mu & = & \diamond \mu \\
 \diamond \mu & = & \diamond \mu & = & \diamond \lambda & \diamond \mu & = & \diamond \lambda & = & \diamond \lambda
 \end{array} \quad (3.24)$$

This little dance shows that multiplication of numbers is furthermore *commutative*. So, what we call ‘numbers’ consists of a set of things that can be ‘multiplied’, and that multiplication operation is associative, commutative, and has a unit (i.e. a chosen number that means ‘1’). Sound familiar? Mathematicians call this a *commutative monoid*. Other people would call this ‘virtually any kind of numbers you can imagine.’

Remark* 3.33 This remarkably simple little proof shown in (3.24) is known as the *Eckmann–Hilton argument*. It shows in general that, whenever we have a pair of associative operations (in this case \otimes and \circ , applied specifically to numbers) that share the same unit (1_I in both cases) and satisfy the interchange law (3.16) (which of course comes for free with diagrams), both of these operations are commutative and are in fact the same operation. Here, we used it to prove that \otimes (or equivalently \circ) gives us a commutative ‘multiplication’ operation for the numbers of any process theory.

So, how should we interpret the fact that, when a state meets an effect, a number pops out? One extremely useful interpretation is that this number gives the *probability* that the effect happens, given the system is in a particular state. Or, in terms of tests, it is the probability that when we test the state ψ for an effect π we get ‘yes’ as the answer:

$$\boxed{
 \begin{array}{l}
 \text{test } \left\{ \begin{array}{c} \triangle \pi \\ \hline \square \\ \hline \nabla \psi \end{array} \right\} \\
 \text{state } \left\{ \begin{array}{c} \triangle \pi \\ \hline \square \\ \hline \nabla \psi \end{array} \right\}
 \end{array}
 \right\} \text{probability} \quad (3.25)$$

We refer to this as the *generalised Born rule*. Here, we say ‘generalised’ because it makes sense in any process theory. We will see in Chapter 6 that by restricting to the theory of **quantum maps**, we obtain the rule that is used to compute all probabilities within quantum theory, which is called simply the *Born rule*.

Remark* 3.34 A reader familiar with quantum theory may think that rather than a probability, in (3.25) we obtain a complex number (a.k.a. an ‘amplitude’), which we then have to multiply with its conjugate in order to obtain a probability. This is indeed the case if we are dealing with plain old **linear maps**. However, **quantum maps** have this step ‘built in’, so the numbers are always positive real numbers. In Example 3.37 below we indicate how (3.25) indeed produces the correct probabilities in quantum theory, and in Section 6.1.1 we explain how the passage from complex numbers to probabilities, via multiplication of a number with its conjugate, is nicely built in to the theory of **quantum maps**.

In many physical theories, and quantum theory in particular, states are thought of as elements in some set (or space), and processes as functions between sets and/or spaces.

If the state is either 0 or 1, then it is *possible* (but no longer certain) that testing for 0 will yield a positive outcome. Finally consider:

$$\begin{array}{c} \triangle \\ | \\ \triangle \end{array} = \emptyset$$

If the system is in the state 0, then it is indeed *impossible* for the test 1 to give a positive outcome.

The following starred example uses some notation that will be introduced over the next two chapters. However, those already familiar with quantum theory may find it useful. Other readers should skip this example for now.

Example* 3.37 (quantum maps) Quantum maps give a particularly elegant way to express completely positive maps, of which mixed quantum states, quantum effects, and probabilities are all special cases. Most of Chapter 6 is devoted to them, but here we provide a first glimpse. For the benefit of those already familiar with quantum theory, we use traditional terminology and notation here, rather than the more process-theoretic versions introduced in Chapter 6.

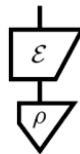
1. *States correspond to density operators.* These are depicted as:



The action of a *completely positive map* \mathcal{E} on a state ρ , that is:

$$\mathcal{E} :: \rho \mapsto \sum_i A_i^\dagger \rho A_i$$

is depicted as:



2. *Effects are positive linear functionals.* These are depicted as:



for some positive operator A . Their action on a state ρ , that is:

$$\rho \mapsto \text{tr}(A\rho)$$

(where tr is the trace) is depicted as:

$$\begin{array}{c} \triangle \\ \text{A} \\ \text{---} \\ \nabla \\ \rho \end{array} \quad (3.26)$$

A special case is the trace itself:

$$\rho \mapsto \text{tr}(\rho)$$

which represents ‘discarding the system’ and is depicted as:



The conditions of being trace-1 (for states) and trace-preserving (for completely positive maps) are written respectively as:

$$\begin{array}{c} \text{---} \\ \text{---} \\ \nabla \\ \rho \end{array} = \text{---} \quad \begin{array}{c} \text{---} \\ \text{---} \\ \square \\ \mathcal{E} \\ \text{---} \\ \text{---} \end{array} = \begin{array}{c} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{array}$$

3. *Numbers are positive reals.* The Born rule (3.25) now takes the form (3.26), and it indeed produces positive reals, which are interpreted as probabilities.

What may surprise the reader is that density matrices are depicted as states, i.e. no input, rather than as input-output processes (cf. density ‘operator’) and that completely positive maps are depicted as input-output processes, rather than as something that takes input-output processes to input-output processes (cf. ‘superoperator’). This is a consequence of the manner in which we define *quantum maps*, and this is of course an upshot, since now the diagrams reflect the true nature of quantum states/effects/processes.

3.4.2 Saying the Impossible: Zero Diagrams

An extreme example of (non-function) relations are *zero relations*. These are relations in which ‘nothing relates to nothing’; i.e. there are no $a \in A$ and $b \in B$ such that $R :: a \mapsto b$, or equivalently:

$$\begin{array}{c} | \\ B \\ \square \\ 0 \\ | \\ A \end{array} :: a \mapsto \emptyset$$

We already encountered a few of these, namely:

$$\begin{array}{c} | \\ \nabla \\ \emptyset \end{array} :: * \mapsto \emptyset \quad \begin{array}{c} \triangle \\ \emptyset \\ | \end{array} :: \begin{cases} 0 \mapsto \emptyset \\ 1 \mapsto \emptyset \end{cases} \quad \emptyset :: * \mapsto \emptyset$$

Each of these represented something ‘impossible’.

Since an effect represents a test in which we successfully verify a certain question, it may be the case that for certain states this effect is simply impossible. So what do we get if we compose that state and that effect in a diagrammatic language? We should of course get a number that corresponds to ‘impossible’. These impossible numbers, and more generally impossible processes, have an elegant characterisation within a process theory.

For the specific case of **relations**, it is easily seen that both the sequential and parallel composition of a zero relation with any relation is again a zero relation. For general process theories, assuming that an effect π is impossible given a state ψ , the number:

$$0 = \begin{array}{c} \triangleup \\ \pi \\ \downarrow \\ \psi \\ \triangleleft \end{array}$$

should represent the impossible. Of course, if part of a larger process is impossible, then the entire process is impossible, hence:

$$\begin{array}{c} | \\ \boxed{0} \\ | \end{array} := 0 \begin{array}{c} | \\ \boxed{f} \\ | \end{array}$$

should also be impossible for any f . So if we want to accommodate the impossible in our language, for every possible input and output type there should be a *zero process*, which obeys the following composition rules:

$$\begin{array}{c} | \\ \boxed{0} \\ | \\ | \\ \boxed{f} \\ | \end{array} = \begin{array}{c} | \\ \boxed{f} \\ | \\ | \\ \boxed{0} \\ | \end{array} = \begin{array}{c} | \\ \boxed{0} \\ | \end{array} \quad \begin{array}{c} | \\ \boxed{0} \\ | \end{array} \begin{array}{c} | \\ \boxed{f} \\ | \end{array} = \begin{array}{c} | \\ \boxed{0} \\ | \end{array} \quad (3.27)$$

Exercise 3.38 Prove that if a zero process satisfying (3.27) exists, then for any given types of input and output systems, it is unique.

So just as multiplying by the plain old number zero always yields zero, zero processes ‘absorb’ any diagram in which they occur. In other words, any diagram containing a zero process is itself a zero process. Due to the uniqueness of the zero process, we will just write it as ‘0’ and ignore its input and output wires:

$$\begin{array}{c} | \\ | \\ | \\ \boxed{h} \\ | \\ | \\ \boxed{0} \quad \boxed{f} \\ | \\ | \\ \boxed{g} \\ | \end{array} = 0$$

The main point to remember here is that 0 is not the empty diagram, since the empty diagram can be adjoined to any diagram without changing it. Zeros, on the other hand, ‘eat’ everything around them!

3.4.3 Processes That Are Equal ‘Up to a Number’

There will be many instances in this book where numbers are crucially important, most notably when the Born rule (3.25) is used to compute probabilities. However, on other occasions, they are just a bit of a nuisance. For example, we may only be interested in some qualitative aspect of a process, such as whether it separates into disconnected pieces (cf. Section 4.1.1 in the next chapter), in which case numbers play no role. In other cases, we can make life a bit easier by ignoring numbers throughout a calculation and figuring out at the end what they should be. Therefore, we’ll introduce some notation to handle these kinds of situations:

Definition 3.39 Two processes are *equal up to a number*, written:

$$\begin{array}{c} | \\ \boxed{f} \\ | \end{array} \approx \begin{array}{c} | \\ \boxed{g} \\ | \end{array}$$

if there exist non-zero numbers λ and μ such that:

$$\lambda \begin{array}{c} | \\ \boxed{f} \\ | \end{array} = \mu \begin{array}{c} | \\ \boxed{g} \\ | \end{array} \quad (3.28)$$

We require λ and μ to be non-zero because otherwise everything would be \approx -related to everything else:

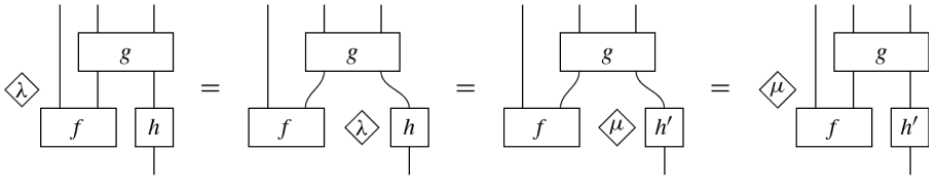
$$0 \begin{array}{c} | \\ \boxed{f} \\ | \end{array} = 0 = 0 \begin{array}{c} | \\ \boxed{g} \\ | \end{array}$$

which is not particularly useful. What should instead be true is that the only thing ≈ 0 is zero itself.

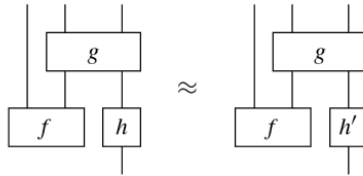
Exercise 3.40 Assume a process theory has *no zero-divisors*; i.e. for all processes f and numbers λ we have $\lambda f = 0$ if and only if $\lambda = 0$ or $f = 0$. Then show that:

$$\begin{array}{c} | \\ \boxed{f} \\ | \end{array} \approx 0 \implies \begin{array}{c} | \\ \boxed{f} \\ | \end{array} = 0$$

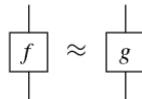
A fortunate feature of the relation \approx is that it plays well with diagrams. Suppose for example that $h \approx h'$, then:



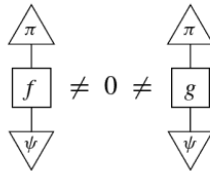
So, for any diagram containing h , we have:



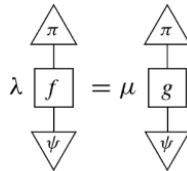
Even if we ignore numbers in a calculation, we might still recover them at the end. Suppose we have established that:



If there exists a state ψ and an effect π such that:



then by sandwiching both f and g in equation (3.28) we obtain:



where all numbers are non-zero. As a consequence, if the numbers of a process theory are not too crazy, then we can figure out what λ and μ must be in order to obtain an equality.

3.4.4 Dirac Notation

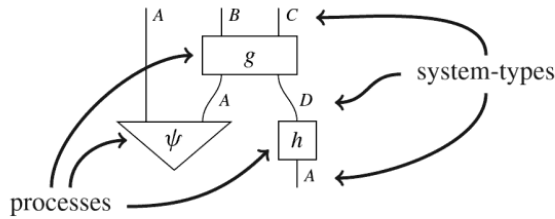
Throughout this chapter we have extolled the virtues of diagrams, most coming from the freedom to write down processes in two dimensions. With modern technology, these diagrams are pretty easy to stick in, say, a textbook. However, things weren't always this good. Let's see now how far we get if we try to turn diagrams into a notation that can easily be churned out on a good old-fashioned typewriter.

Of course, this is ambiguous unless we already know that we are working with some system $A \otimes B$, and pretty soon we need to start requiring extra equations, as in Section 3.2.4.

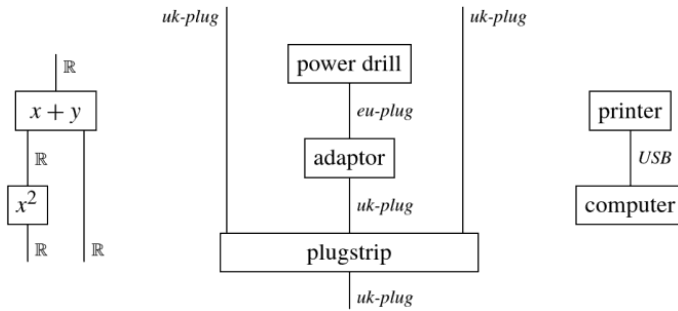
By sticking to diagrams all of this is immediately solved, and fortunately, very few of us are still using typewriters. Technology marches on, and now there are some great tools for drawing diagrams (pretty much) as easily as typing out a Dirac-style formula on a keyboard.

3.5 Summary: What to Remember

1. *Diagrams* consist of boxes that are labelled by *processes* and wires that are labelled by *system-types*:

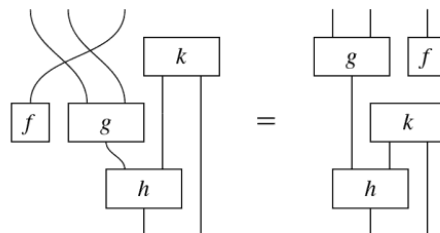


Diagrams are all around us:

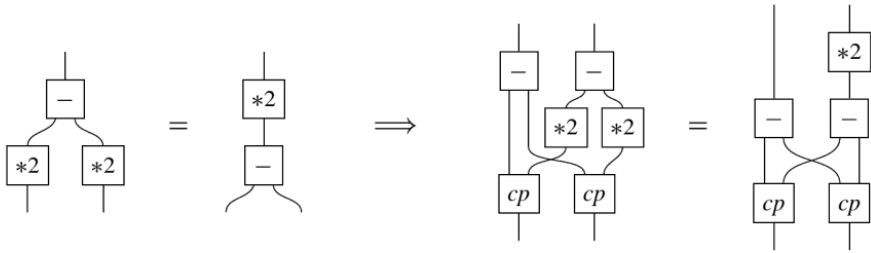


We can obtain new *diagram equations* from old ones in two ways:

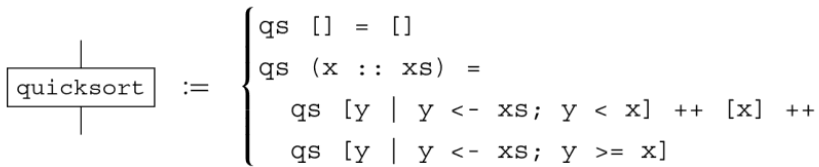
1. *Diagram deformation*:



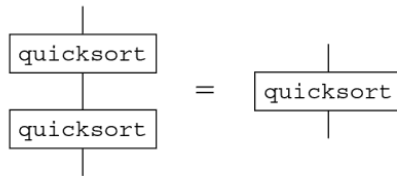
2. Diagram substitution:



2. A *process theory* is a collection of processes that can be plugged together. It tells us how to interpret the boxes and wires in a diagram, e.g.:

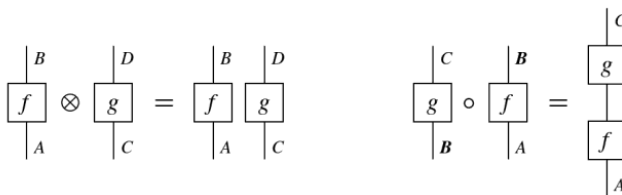


and what it means to ‘wire processes together’. By doing so it also tells us which diagrams represent the same process, e.g.:



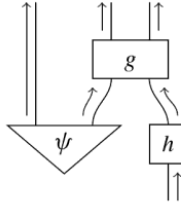
Examples are **functions**, **relations**, **linear maps** (defined in Chapter 5), **classical processes** (defined in Chapter 8), **quantum maps** (defined in Chapter 6), and **quantum processes** (also defined in Chapter 8).

3. *Parallel composition* ‘ \otimes ’ and *sequential composition* ‘ \circ ’ are defined as:



4. *Circuit diagrams* are an important class of diagrams. They have the following two equivalent characterisations:

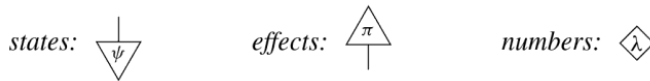
The diagram contains no directed cycles. That is, information flows from bottom to top without ‘feeding back’ to an earlier process:



The diagram can be constructed by composing boxes, including identities and crossings, by means of the operations \otimes and \circ :

$$\left(\left(\text{identity} \right) \otimes \left(\text{box } g \right) \right) \circ \left(\left(\text{box } \psi \right) \otimes \left(\text{box } h \right) \right)$$

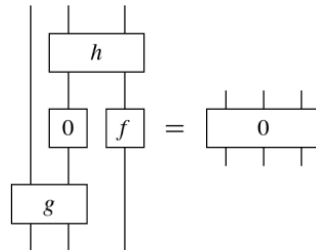
5. Processes lacking inputs, outputs, or both have special names:



These special processes admit a *generalised Born rule*:

$$\left. \begin{array}{l} \text{effect } \left\{ \begin{array}{c} \triangle \pi \\ \mid \\ \triangle \psi \end{array} \right\} \\ \text{state } \left\{ \begin{array}{c} \triangle \pi \\ \mid \\ \triangle \psi \end{array} \right\} \end{array} \right\} \text{number}$$

6. Zero processes ‘eat everything’:



7. Dirac notation is common language for depicting quantum processes. It is a fragment of the diagrammatic language:

$$\begin{array}{ccc} \triangle \psi \leftrightarrow |\psi\rangle & \triangle \pi \leftrightarrow \langle \pi| & \begin{array}{c} \triangle \pi \\ \mid \\ \triangle \psi \end{array} \leftrightarrow \langle \pi|\psi\rangle \end{array}$$

3.6 Advanced Material*

Central to this book are process theories, which tell us how to interpret wires and boxes in a diagram, and what it means to form diagrams, i.e. what it means to wire boxes together. Here we present two alternative ways of defining process theories that make no direct reference to diagrams, namely, *abstract tensor systems* and *symmetric monoidal categories*.

In fact, we already saw a glimpse of each of these, respectively, when we encountered diagram formulas in Section 3.1.3 and when we defined circuits by means of the operations parallel and sequential composition in Section 3.2.

So how does one go about constructing a symbolic counterpart to the definition of process theories? In the first two parts of the definition:

- (i) a collection T of *system-types* represented by wires,
- (ii) a collection P of *processes* represented by boxes, where for each process in P the input types and output types are taken from T ...

diagrams don't play an essential role and we can just drop the 'represented by ...' bit, provided that we explicitly require that each process comes with a list of input types and a list of output types, all taken from T . The third part of the definition, which concerns what 'forming diagrams' means:

- (iii) a means of 'wiring processes together', that is, an operation that interprets a diagram of processes in P as a process in P

is the one that requires some non-trivial effort. In particular, we need to provide a symbolic counterpart to 'wiring processes together' that does not make reference to diagrams. Moreover, the symbolic operation(s) should produce processes of that theory.

3.6.1 Abstract Tensor Systems*

We introduced diagram formulas as a symbolic counterpart to diagrams. Can we define a corresponding notion of process theory? The definition below does exactly that. For notational simplicity we will assume that all system-types are the same, and consequently, distinct wire names are merely distinguishing occurrences of the same system.

Definition 3.42 An abstract tensor system (ATS) consists of:

1. For all wire names A_1, \dots, A_m and B_1, \dots, B_n , a set of *tensors*:

$$f_{A_1 \dots A_m}^{B_1 \dots B_n} \in \mathcal{T}(\{A_1, \dots, A_m\}, \{B_1, \dots, B_n\})$$

where A_1, \dots, A_m are called *inputs* and B_1, \dots, B_n are called *outputs*.

2. A *unit* tensor:

$$1 \in \mathcal{T}(\{\}, \{\})$$

and for all wire names A and B *identity* tensors:

$$\delta_A^B \in \mathcal{T}(\{A\}, \{B\})$$

3. An operation *tensor product* that combines tensors provided their inputs and outputs are distinct, which is denoted by concatenation:

$$f_{A_1 \dots A_m}^{B_1 \dots B_n} g_{C_1 \dots C_k}^{D_1 \dots D_l} \in \mathcal{T}(\{A_1, \dots, A_m, C_1, \dots, C_k\}, \{B_1, \dots, B_n, D_1, \dots, D_l\})$$

4. For any wire names A and B , an operation c_A^B called *tensor contraction*, which is denoted either explicitly or by repeating an upper/lower wire name:

$$f_{A_1 \dots A_m}^{B_1 \dots B_{j-1} A_i B_{j+1} \dots B_n} := c_{A_i}^{B_j} \left(f_{A_1 \dots A_m}^{B_1 \dots B_n} \right)$$

which, intuitively, ‘connects’ output B_j to input A_i .

5. *Re-indexing* operations that change wire names:

$$f_{A_1 \dots A_m}^{B_1 \dots B_n} [A_i \mapsto A'_i] = f_{A_1 \dots A_{i-1} A'_i A_{i+1} \dots A_m}^{B_1 \dots B_n}$$

where these operations satisfy the following conditions:

1. the tensor product is associative, commutative, and has unit 1:

$$(fg)h = f(gh) \qquad fg = gf \qquad 1f = f$$

2. the order of tensor contractions/products is irrelevant:

$$c_A^B(c_C^D(f)) = c_C^D(c_A^B(f)) \qquad c_A^B(f)g = c_A^B(fg)$$

3. contracting with the identity does nothing except change wire names:

$$c_A^B(\delta_C^B f \dots) = f \dots C \dots \qquad c_B^A(\delta_B^C f \dots) = f \dots C \dots$$

4. re-indexing respects identities, tensor product, and contraction.

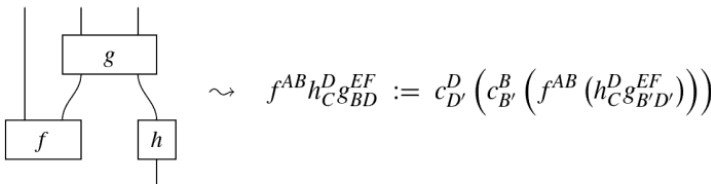
The two operations, tensor product and tensor contraction, together play the role of ‘wiring processes together’. For example, the symbolic counterpart to the diagram:



is obtained as follows:

$$c_{B'}^B \left(f_A^B g_{B'}^C \right)$$

More generally, we can compute the tensor corresponding to a diagram by first writing it down as a diagram formula and then decomposing it further in terms of the tensor product and tensor contraction. For example:



Definition 3.47 An object A is *isomorphic* to an object B , denoted:

$$A \cong B$$

if and only if there exists a pair of morphisms:

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

such that:

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

The morphism f is then called an *isomorphism*.

Most monoidal categories we find in the wild tend to be of the non-strict variety. That is, their parallel composition only satisfies:

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

We already briefly encountered this situation in Section 3.3.1. Recall that \otimes was defined in terms of the Cartesian product. When we compare the sets $(A \times B) \times C$ and $A \times (B \times C)$ (and the sets A and $A \times \{*\}$), we do not get *equal* sets, but rather sets whose elements are in one-to-one correspondence:

$$((a, b), c) \leftrightarrow (a, (b, c)) \qquad a \leftrightarrow (a, *)$$

To define a non-strict monoidal category, we must include these correspondences in the definition of the category as so-called *structural isomorphisms*, and we must assume several (more!) equations called *coherence equations* that guarantee that they behave sensibly when they are composed together.

However, we happily glazed over this fact when we defined the process theories of functions and relations in Section 3.3. Why? Because of the following *coherence theorem*.

Theorem 3.48 Every (symmetric) monoidal category \mathcal{C} is equivalent to a strict (symmetric) monoidal category \mathcal{C}' .

Equivalent categories are, for all practical purposes, the same. We'll say more about what this means in Section* 5.6.4. The proof of Theorem 3.48 consists of an explicit construction of the category \mathcal{C}' by a procedure called *strictification*. This procedure is used implicitly when, e.g. we decide to treat elements $((a, b), c)$, $(a, (b, c))$ and (a, b, c) as 'the same', which we do without further comment throughout this text.

Furthermore, we are justified in using circuit diagrams to talk about symmetric monoidal categories because of the following theorem.

Theorem 3.49 Circuit 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 circuit diagram.

To summarise, it is possible to translate many concepts between our purely diagrammatic language and category theory, using this correspondence:

process theory \leftrightarrow (strict) symmetric monoidal category
 process \leftrightarrow morphism
 system-type \leftrightarrow object

3.6.3 General Diagrams versus Circuits*

Some readers may have noted that abstract tensor systems and symmetric monoidal categories aren't exactly one and the same. While symmetric monoidal categories correspond to circuits, abstract tensor systems represent more general kinds of diagrams in which directed cycles (cf. Definition 3.20) are allowed. However, we can make the two concepts coincide.

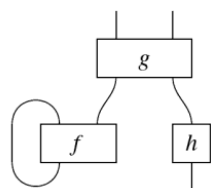
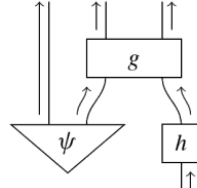
In one direction, one can force abstract tensor systems to be equivalent to strict symmetric monoidal categories by only requiring the contraction operation c_A^B to be well-defined if it does not introduce a directed cycle.

Conversely, it is possible to introduce 'feedback loops' into symmetric monoidal categories, obtaining so-called *traced symmetric monoidal categories*. These categories assume an additional operation called the *trace*:

$$\begin{array}{c} \text{C} \\ | \\ \text{---} \text{---} \text{---} \\ | \\ \text{A} \end{array} \text{---} \begin{array}{|c|} \hline f \\ \hline \end{array} \begin{array}{c} \text{---} \text{---} \text{---} \\ | \\ \text{B} \end{array} \quad := \quad \text{tr}_A \left(\begin{array}{|c|c|} \hline \text{A} & \text{C} \\ \hline \text{---} \text{---} \text{---} & \text{---} \text{---} \text{---} \\ \hline \text{A} & \text{B} \\ \hline \end{array} \right)$$

which we will meet in Section 4.2.3 of the next chapter. The trace satisfies a handful of axioms to ensure that it behaves like a 'feedback loop' and that it respects the other categorical structure. Then, strict traced symmetric monoidal categories are equivalent to abstract tensor systems, in that it is possible to turn an abstract tensor system into a strict traced symmetric monoidal category and vice versa without losing any meaningful information.

We can summarise the above in the following table:

	'General' diagram	Circuit diagram
Diagrams	 <p style="text-align: center;">i.e. outputs to inputs</p>	 <p style="text-align: center;">i.e. admits causal structure</p>
ATS	ATS	ATS without cycles
SMC	traced SMC	SMC

3.7 Historical Notes and References

The idea of representing the composition of mathematical objects using nodes and wires dates back a very long time. A prominent example is the flow chart, which goes back (at least) to the 1920s (Gilbreth and Gilbreth, 1922). In physics, the most notable example is Feynman diagrams. Despite their name, it is now recognised that they originated in the work of Ernst Stueckelberg around 1941 and were only later independently re-derived by Richard Feynman around 1947. In a lecture at CERN after having been awarded the Nobel Prize Richard Feynman referred to this fact the moment that Stueckelberg left the room: ‘He did the work and walks alone toward the sunset; and, here I am, covered in all the glory, which rightfully should be his!’ (Mehra, 1994).

Penrose (1971) was the first to introduce the particular kind of diagrams we use in this book as an alternative to the abstract tensor notation (see Section* 3.6.1), which he also introduced. In fact, he already started using diagrams when he was still an undergraduate student (Penrose, 2004).

Paul Dirac introduced his new notation in Dirac (1939). The connection between Dirac notation and the corresponding use of the triangle notation for states and effects in string diagrams was observed in Coecke (2005). The idea of treating diagrams themselves as the basic structure rather than categories has recently become prominent in the quantum foundations community. It has been used, for example, in efforts towards crafting alternative formulations of quantum theory and a theory of quantum gravity (Chiribella et al., 2010; Coecke, 2011; Hardy, 2011, 2013b).

The unwillingness of many to accept diagrams as a rigorous mathematical language (alluded to in Section 3.1.3) may be largely due to the Bourbakimindset in which mathematics has been taught for much of the previous century. The Bourbaki collective aimed to found all mathematics on a set-theoretic basis (Bourbaki, 1959–2004). Great credit should be given to John Baez, the pioneer of scientific blogging, who by means of his outstanding didactical skills managed to convince many of the amazingness of diagrams. His old weekly column ‘This Week’s Finds in Mathematical Physics’ is still a great resource on diagrammatic reasoning (Baez, 1993–2010).

The connection between circuit diagrams and algebraic structure (in the form of symmetric monoidal categories, cf. Theorem 3.49) was established in Joyal and Street (1991), where circuit diagrams are referred to as ‘progressive polarised diagrams’. Traced symmetric monoidal categories and their graphical notation were introduced in Joyal et al. (1996); however, the closely related notion of a ‘compact closed category’ (whose diagrams are string diagrams, which we will encounter very soon) had already been known for some time. The historical notes in the next chapter provide appropriate references.

The proof of correctness of the graphical language for traced symmetric monoidal categories has been sketched in various papers (e.g. Hasegawa et al., 2008), and a full proof is given by Kissinger (2014a). Following on from Joyal and Street’s work came a plethora of variations of these diagrams and corresponding algebraic structures (see Selinger, 2011b).

Monoidal categories themselves were first introduced in Benabou (1963). Theorem 3.48 on coherence is crucial for connecting monoidal categories to diagrams and was first stated and proved by Mac Lane (1963). The ‘little dance’ in (3.24), a.k.a. the Eckmann–Hilton argument, is from Eckmann and Hilton (1962).

Abstract tensor systems, and their associated ‘abstract index notation’, were introduced by Penrose (1971) at the same time as the corresponding diagrammatic notation, in order to talk about various kinds of multilinear maps without needing to fix bases in advance. This notation is now very common in theoretical physics and especially general relativity. Penrose provides an introduction for a general audience in Penrose (2004). The equivalence between abstract tensor systems and traced symmetric monoidal categories is given by Kissinger (2014a).

If you want to read more on general category theory, some good places to start are Abramsky and Tzevelekos (2011) for the connection between categories and logic, Coecke and Paquette (2011) for the generality of the notion of symmetric monoidal categories, and Baez and Lauda (2011) for the role categorical structures have played in physics. Standard textbooks for mathematicians are Borceux (1994a,b) and Mac Lane (1998); standard textbooks for computer scientists are Barr and Wells (1990) and Pierce (1991); and standard textbooks for logicians are Lambek and Scott (1988) and Awodey (2010). In the context of category theory, the idea that morphisms represent processes emerged mostly in the context of applications in computer science, where programs are represented by morphisms in a category and data types are the objects. Similarly in logic, proofs can be represented by morphisms in a category, with propositions as the objects. This trifecta of morphisms, programs, and proofs are all connected by what’s known as the *Curry–Howard–Lambek isomorphism* (Lambek and Scott, 1988).

The quote at the beginning of this chapter is taken from Bohm and Peat (1987). As discussed in Section 1.3, Bohm’s views played a key role in making process ontology more prominent in physics.

4

String Diagrams

When two systems, of which we know the states by their respective representatives, enter into temporary physical interaction due to known forces between them, and when after a time of mutual influence the systems separate again, then they can no longer be described in the same way as before, viz. by endowing each of them with a representative of its own. I would not call that one but rather the characteristic trait of quantum mechanics, the one that enforces its entire departure from classical lines of thought.

– Erwin Schrödinger, 1935

By 1935, Schrödinger had already realised that the biggest gulf between quantum theory and our received classical preconceptions is that, when it comes to quantum systems, the whole is more than the sum of its parts. In classical physics, for instance, it is possible to totally describe the state of two systems – say, two objects sitting on a table – by first totally describing the state of the first system then totally describing the state of the second system. This is a fundamental property one expects of a classical, *separable* universe. However, as Schrödinger points out, there exist states predicted by quantum theory (and observed in the lab!) that do not obey this ‘obvious’ law about the physical world. Schrödinger called this new, totally non-classical phenomenon *Verschränkung*, which later became translated to the dominant scientific language as *entanglement*.

Quantum picturalism is all about studying the way parts compose to form a whole. In the good company of Schrödinger, we believe that the role of multiple interacting systems – especially non-separable systems – should take centre stage in the study of quantum theory.

We shall see in the next section that it is easy to say what it means for a process to be *separable* in terms of diagrams. Literally, it means that it can be broken up into pieces that are not connected to each other. On the other hand, enforcing *non*-separability requires us to refine our diagrammatic language. To this end, we introduce special states and effects called *cups* and *caps*, respectively. Intuitively, cups and caps act like pieces of wire that have been ‘bent sideways’. Whereas all states in the classical world

$$\begin{array}{c} \downarrow \quad \downarrow \\ \triangle C \\ \uparrow \quad \uparrow \end{array} = \begin{array}{c} \downarrow \quad \downarrow \\ \triangle B \quad \triangle B' \\ \uparrow \quad \uparrow \end{array}$$

for some $B, B' \subseteq \mathbb{B}$. Then, by (3.22):

$$\begin{array}{c} \downarrow \quad \downarrow \\ \triangle C \\ \uparrow \quad \uparrow \end{array} :: * \mapsto (b, b') \iff \begin{array}{c} \downarrow \\ \triangle B \\ \uparrow \end{array} :: * \mapsto b \text{ and } \begin{array}{c} \downarrow \\ \triangle B' \\ \uparrow \end{array} :: * \mapsto b'$$

that is:

$$(b, b') \in C \iff b \in B \text{ and } b' \in B'$$

Since $(0, 0) \in C$, it follows that $0 \in B$. Similarly, $(1, 1) \in C$ implies that $1 \in B'$. But these two facts imply $(0, 1) \in C$, which is a contradiction. \square

The crux of this proof is the fact that the (non-singleton) set:

$$\{(0, 0), (1, 1)\}$$

cannot be written as a Cartesian product of subsets of \mathbb{B} . If as in **functions**, every such set is a singleton, then this is always possible.

A closely related notion to \otimes -separability is the following.

Definition 4.5 We call a process:



o-separable if there is an effect π and a state ψ such that:

$$\begin{array}{c} \downarrow \\ \square f \\ \uparrow \end{array} = \begin{array}{c} \downarrow \\ \triangle \psi \\ \uparrow \\ \triangle \pi \\ \uparrow \end{array} \tag{4.3}$$

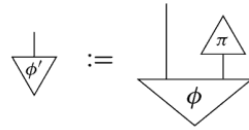
Both \otimes -separability of bipartite states and o-separability of processes are examples of ‘disconnectedness’ of the corresponding diagrams. Whenever no confusion is possible we will simply say ‘separable’ in both cases.

Moreover, these two notions of separability are related to each other in the following sense.

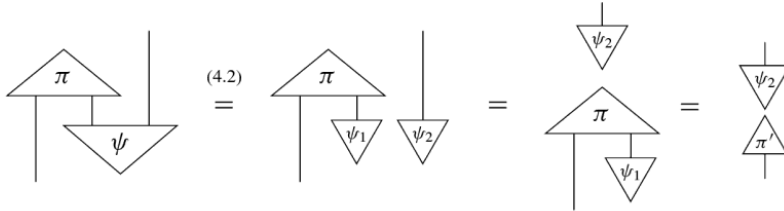
- If the process f is o-separable, then for any bipartite state ϕ the following state is \otimes -separable:

$$\begin{array}{c} \downarrow \quad \downarrow \\ \square f \\ \uparrow \quad \uparrow \\ \triangle \phi \end{array} \stackrel{(4.3)}{=} \begin{array}{c} \downarrow \quad \downarrow \\ \triangle \psi \\ \uparrow \\ \triangle \pi \\ \uparrow \\ \triangle \phi \end{array} = \begin{array}{c} \downarrow \quad \downarrow \\ \triangle \pi \\ \uparrow \\ \triangle \phi \end{array} \downarrow \psi = \begin{array}{c} \downarrow \\ \triangle \phi' \\ \uparrow \end{array} \begin{array}{c} \downarrow \\ \triangle \psi \\ \uparrow \end{array}$$

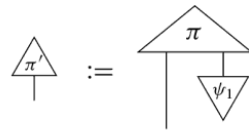
where:



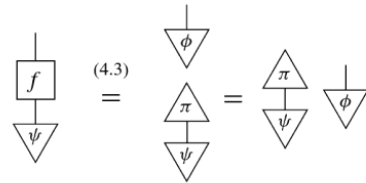
- If the state ψ is \otimes -separable, then for any bipartite effect π the following process is \circ -separable:



where:



We also saw earlier that in **functions** all bipartite states are \otimes -separable. So what kinds of process theories have all \circ -separable processes? The answer is, ‘only the really boring ones!’ When we apply a \circ -separable process to an arbitrary state:



we cannot obtain anything other than a fixed state ϕ (ignoring the number $\pi \circ \psi$), regardless of what the input ψ was. That is, every process is essentially a *constant* process. In other words:

nothing ever happens!

Thus it shouldn't be surprising that the theory of **functions** *does* contain \circ -non-separable processes.

Exercise 4.6 Show that identities (i.e. plain wires) in **functions** are \circ -non-separable, and characterise those functions that are \circ -separable.

Remark 4.7 The ultimate boringness of process theories where all processes are \circ -separable will play an important role in several ‘no-go’ theorems that we develop later in this chapter. In particular, we will show that some statement P is not true by proving a theorem of the form:

If P is true, then all processes are \circ -separable.

As this is an absurd condition for any reasonable process theory, and in particular any theory of physics, we can take this as a proof that P is false.

4.1.2 Process–State Duality

One feature of quantum theory is that one can turn a process into a bipartite state and vice versa. This is in itself not very significant: we already demonstrated such a conversion in the previous section in order to relate \otimes -separability of bipartite states to \circ -separability of processes. What is significant about quantum theory is that this can be done in a reversible manner. That is, we have a way to turn a process into a bipartite state and then back into a process such that we obtain the original process, and vice versa. In other words, the operations that send processes to bipartite states and bipartite states to processes are inverses of each other. Therefore, in quantum theory, processes and bipartite states are in bijective correspondence.

As in the previous section, we will make use of bipartite states and effects to inter-convert processes and states. However, we will not be using just any state/effect pair but instead fix a special state and effect:

$$\begin{array}{c} |A \quad |A \\ \hline \nabla \\ U \end{array} \quad \text{and} \quad \begin{array}{c} \nabla \\ \hline |A \quad |A \end{array} \quad (4.4)$$

for each system A . We call these states and effects *cups* and *caps*, respectively. They can be used to convert processes to states and back via the following operations:

$$\begin{array}{c} | \\ \square \\ f \\ | \end{array} \xrightarrow{(*)} \begin{array}{c} | \\ | \\ \square \\ f \\ | \\ \nabla \\ U \end{array} \quad \begin{array}{c} | \\ \nabla \\ \psi \end{array} \xrightarrow{(\diamond)} \begin{array}{c} \nabla \\ | \\ \nabla \\ \psi \end{array} \quad (4.5)$$

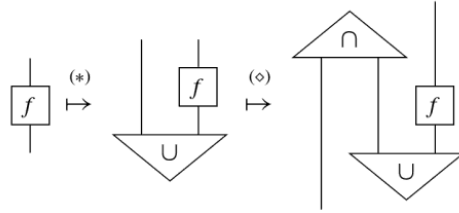
Theorem 4.8 The operations given in (4.5) are inverses of each other, i.e. they give *process–state duality*, if and only if:

$$\begin{array}{c} \nabla \\ | \\ \nabla \\ U \end{array} \stackrel{(a)}{=} \begin{array}{c} | \\ \nabla \\ U \end{array} \quad \begin{array}{c} \nabla \\ | \\ \nabla \\ \psi \end{array} \stackrel{(b)}{=} \begin{array}{c} \nabla \\ \psi \end{array} \quad (4.6)$$

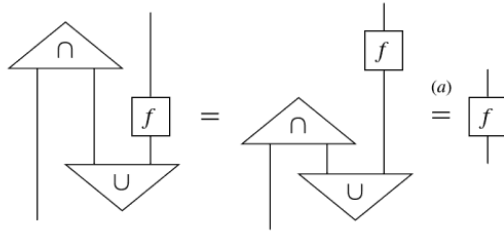
In that case, the set of all processes with input type A and output type B is in one-to-one correspondence with the set of all bipartite states of type $A \otimes B$:

$$\left\{ \begin{array}{c} |B \\ \square \\ f \\ |A \end{array} \right\}_f \cong \left\{ \begin{array}{c} |A \quad |B \\ \nabla \\ \psi \end{array} \right\}_\psi$$

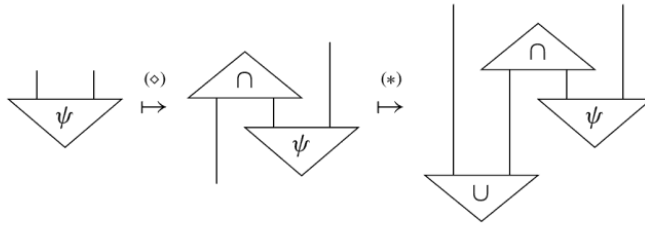
Proof First, assume equations (a) and (b). Then, if we first turn a process into a state and back:



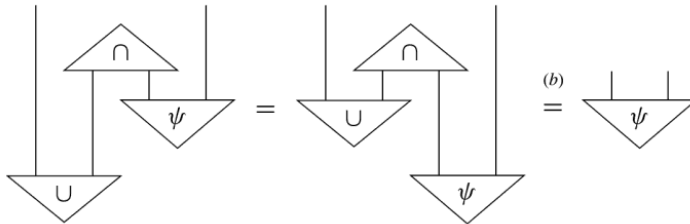
we can simplify back to the original process using (a):



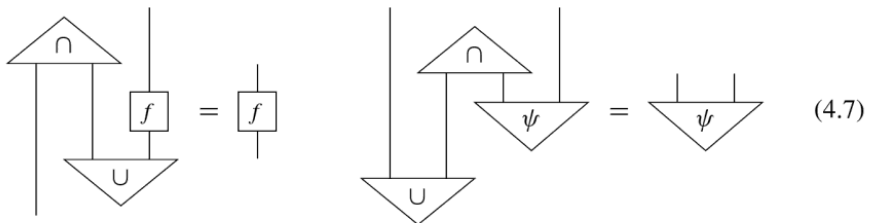
Similarly, starting with a state:



we can use (b):

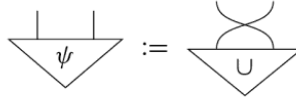


Conversely, if the operations given in (4.5) are mutually inverse, then:



for all f and ψ . Taking f to be the identity yields (a) immediately. Proving (b) using the above equations is left as an exercise. \square

Exercise 4.9 Prove equation (4.6b) using both of the process–state duality equations (4.7), by setting:



Exercise 4.10 Show that process–state duality does not hold for **functions**, but that it does hold for **relations**.

Remark* 4.11 A reader familiar with quantum theory may know about a correspondence between processes and states called the *Choi–Jamiołkowski isomorphism*. Process–state duality is in fact a generalisation of the C–J isomorphism. In Chapter 6, we will introduce the process theory of **quantum maps**, in which case this generalised C–J isomorphism will actually be the familiar correspondence between bipartite (possibly mixed) quantum states and completely positive maps.

4.1.3 The Yanking Equations

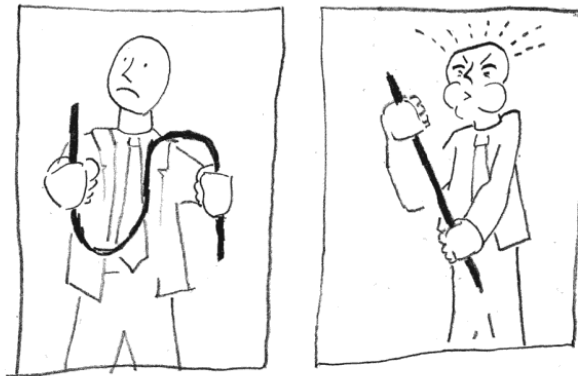
Equation (4.6) is not very intuitive when viewed diagrammatically, but by means of a simple change of notation we can fix this. We write the special states and the special effects, respectively, as U-shaped and \cap -shaped wires:



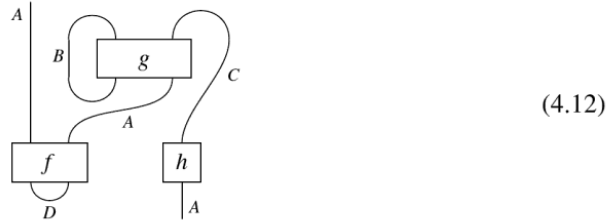
Equation (4.6) now becomes:

$$\text{U-shaped wire} = | = \text{cap-shaped wire} \tag{4.8}$$

This notation expresses the fact that bent wires involving a U-shape and a \cap -shape can be yanked into a straight wire:



Definition 4.18 A *string diagram* consists of boxes and wires, where we additionally allow inputs to be connected to inputs and outputs to be connected to outputs, for example:



Above we already indicated that we can replace the special states and the special effects of the previous section by cup-shaped wires and by cap-shaped wires, respectively. This is how we obtain a string diagram from a circuit diagram in which there are states and effects satisfying (4.11). Conversely, starting from a string diagram we can replace a wire connecting two inputs with the special cup state and a wire connecting two outputs with the special cap effect. Hence the following theorem.

Theorem 4.19 The following two notions are equivalent:

- (i) string diagrams and
- (ii) circuit diagrams to which we adjoin a special state and a special effect for each type, and for which (4.11) holds.

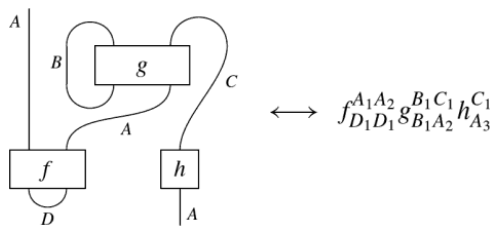
in the sense that (ii) can be unambiguously expressed as (i) and vice versa.

Bingo! We expressed non-separability in purely diagrammatic terms, and not just by adding some new boxes to (the now boring) circuit diagrams, but also by simply considering a different, more liberal kind of diagram. Thus, the most important feature of quantum theory (according to Schrödinger) is built right into the kinds of diagrams we use!

Exercise 4.20 Write diagram (4.12) in \circ and \otimes notation using cups and caps.

String diagrams also admit a formula-like presentation (cf. Section 3.1.3).

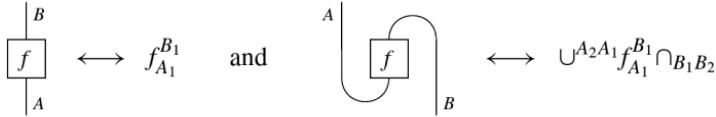
Definition 4.21 A *string diagram formula* is the same as a diagram formula except for the fact that matched pairs of wire names, besides consisting of an upper and a lower name, can now also consist either of two upper names or of two lower names, for example:



Remark 4.22 In Section 3.1.3, we introduced special box names $1_{A_1}^{A_2}$ to represent plain wires in diagram formulas. If we need explicit cups or caps in string diagram formulas, we can use two ‘plain wires’ with their inputs or outputs, respectively, connected together:

$$\cup^{A_1 A_2} := 1_{A_3}^{A_1} 1_{A_3}^{A_2} \qquad \cap_{A_1 A_2} := 1_{A_1}^{A_3} 1_{A_2}^{A_3}$$

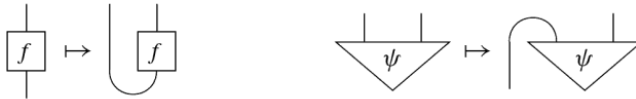
This allows us to distinguish, for example:



We now explore the richness of these diagrams.

4.2 Transposition and Trace

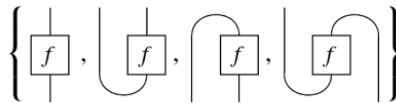
Transposition and trace are notions you may know from linear algebra. Quite remarkably, these already arise at the very general level of string diagrams. We already know from the previous section that cups and caps let us form new processes from old ones by turning inputs into outputs and vice versa. In the case of process–state duality we have:



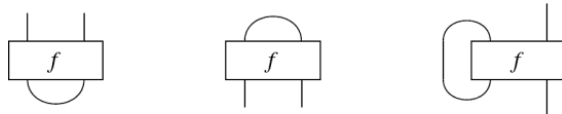
More generally, we can always inter-convert inputs and outputs:



This ability to inter-convert inputs and outputs for process theories that admit string diagrams means that inputs and outputs have a less profound status than in the case of other process theories that only admit circuits. In particular, any process with an input and an output has (at least) the following four inter-convertible representations:



and in the case of several inputs and outputs there are many more. We also can obtain new processes by connecting inputs and outputs together using cups and caps:

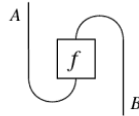


The *transpose* and the *trace* are the most prominent examples where caps and cups are used to convert processes into new processes.

4.2.1 The Transpose

In a process theory that admits string diagrams we can associate to each process another process going in the opposite direction.

Definition 4.23 The *transpose* f^T of a process f is another process:



This of course should not be confused with another well-known way to obtain a process going in the other direction:

Definition 4.24 A process f from type A to type B has an *inverse* if there exists another process f^{-1} from type B to type A such that:

$$\begin{array}{c} A \\ | \\ \boxed{f^{-1}} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array} = \left| \begin{array}{c} A \\ | \\ \boxed{f^{-1}} \\ | \\ B \\ | \\ \boxed{f} \\ | \\ A \end{array} \right. \qquad \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \\ | \\ \boxed{f^{-1}} \\ | \\ B \end{array} = \left| \begin{array}{c} B \\ | \\ \boxed{f} \\ | \\ A \\ | \\ \boxed{f^{-1}} \\ | \\ B \end{array} \right. \qquad (4.13)$$

Exercise 4.25 Show that if a process f has an inverse, it is unique.

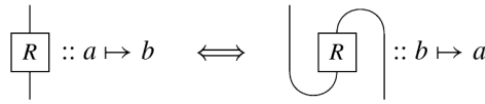
While the transpose can be realised by a diagram involving f :

$$\begin{array}{c} \text{---} \\ | \\ \text{---} \\ | \\ \boxed{f} \\ | \\ \text{---} \\ | \\ \text{---} \end{array} = (1_A \otimes \cap_B) \circ (1_A \otimes f \otimes 1_B) \circ (\cup_A \otimes 1_B) \qquad (4.14)$$

this is not the case for an inverse, otherwise every process would have an inverse, which is usually not true.

Remark* 4.26 The simple fact that the transpose in linear algebra can be written using a cup and a cap as in decomposition (4.14) is surprisingly not very well known, even among specialists.

Exercise 4.27 Prove that in **relations** the transpose of a relation R is the converse relation, that is:



A state:



has no input, so we only have to convert the output into an input:

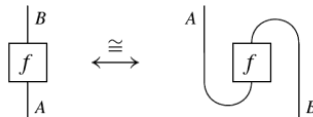


Hence, it is no coincidence that in **relations** any system-type A has the same number of states as effects (see Example 3.36), since we have a bijective correspondence between states and effects:



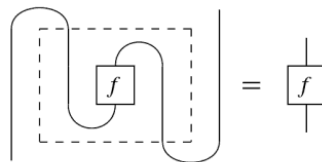
This is a general phenomenon.

Proposition 4.28 For any process theory that admits string diagrams there is a bijective correspondence between states and effects of the same type. More generally, the correspondence:



yields a bijective correspondence between the processes of a fixed input and output type, and the processes with the opposite input and output type.

The transpose of a transpose yields the original process:



Or, symbolically:

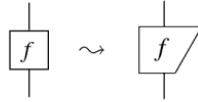
$$(f^T)^T = f$$

An operation that ‘undoes itself’ like this is called an *involution*.

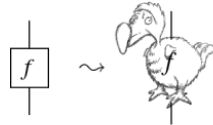
Example 4.29 The transpose of a cup is a cap and vice versa:



Now comes the really cool bit. We can build the definition of transpose into our diagrammatic notation. First, we deform our boxes a bit:



It doesn't matter too much how we deform boxes, only that we break the symmetry. For example,



would also work, but Dave might not appreciate being skewered. Now, we express the transpose of f as a box labelled ' f ', but rotated 180°:

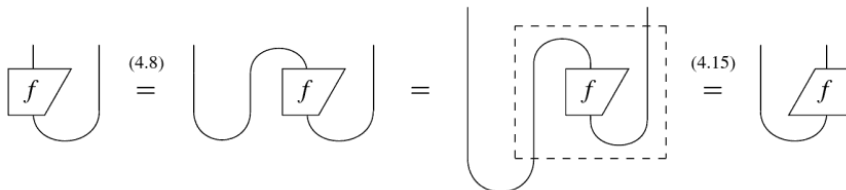
$$\begin{array}{c} \diagup \\ f \\ \diagdown \end{array} := \begin{array}{c} \diagdown \\ f \\ \diagup \end{array} \quad (4.15)$$

This notation is clearly consistent with the fact that the transpose is an involution, since, if one rotates by 180° twice, one obtains the original box again. However, the real advantage of this choice of notation comes when transposes interact with cups and caps.

Proposition 4.30 For any process f we have:



Proof The first equality is established as follows:

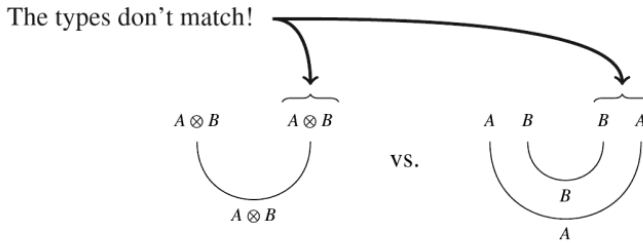


and the proof of the second one proceeds similarly. □

$$\begin{array}{|c|} \hline B \\ \hline \hline A \\ \hline \hline D \\ \hline \hline C \\ \hline \end{array} \xrightarrow{f} \begin{array}{|c|} \hline C \\ \hline \hline D \\ \hline \hline A \\ \hline \hline B \\ \hline \end{array}$$

$$\equiv \text{Diagram with strands } A, B, C, D \text{ and crossings}$$
(4.19)

This would suggest that we should ‘nest’ the caps and cups inside of each other to define caps and cups for $A \otimes B$. However, there’s a problem with defining $\cup_{A \otimes B}$ and $\cap_{A \otimes B}$ as above:



Remark* 4.32 This type mismatch vanishes if we introduce for every type A a *dual type* A^* , but this is at the cost of having two distinct types for the two systems involved in cups/caps, while in fact, these two types are often essentially the same. In Section* 4.6.2 we show how one can develop a theory of string diagrams with dual types.

We can avoid this type mismatch if we define cups/caps for $A \otimes B$ a bit differently, namely as ‘criss-crossed’ cups/caps:

$$A \otimes B \text{ (cup)} := A \text{ } B \text{ (cup)} \text{ } A \text{ } B \text{ (cup)}$$
(4.20)

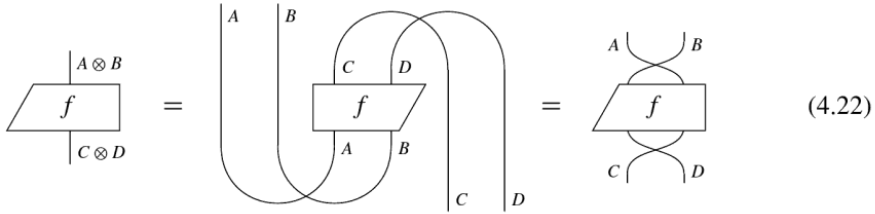
$$A \otimes B \text{ (cap)} := A \text{ (cap) } B \text{ (cap) } A \text{ (cap) } B \text{ (cap)}$$
(4.21)

Proposition 4.33 The cup and cap defined in equations (4.20) and (4.21) satisfy the yanking equations (4.11).

Proof For the first yanking equation of (4.11) we have:

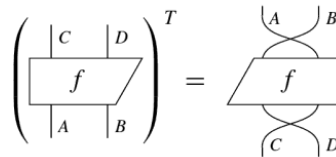
Proofs of the other two equations are similar. □

With these ‘criss-crossed’ cups/caps, we obtain an alternative notion of transposition, which introduces a twist for composite systems:



It is this twist that restores the matching of types, and since we define cups/caps on $A \otimes B$ in terms of cups/caps on A and B individually, there is no ambiguity when we apply this transpose on processes involving composite systems. But, at the same time, we lost a chunk of the elegant ‘180° rotation’ notation for transposition.

In fact, it turns out that both versions of transposition can be useful. We will take the first version, indicated in (4.19), to be the default and continue to refer to this simply as ‘the transpose’. The second version, involving the ‘criss-crossed’ cups and caps, will be referred to as the *algebraic transpose*, since it is in fact the one that is used in linear algebra (see Section 5.2.2). While the transpose is (still) denoted by 180° rotation, we will use the symbolic notation $()^T$ to represent the algebraic transpose. So the equation:



now relates the transpose to the algebraic transpose. Of course, when a box has at most one input/output wire, these two versions coincide. Unsurprisingly, a process that is equal to its algebraic transpose is called *algebraically self-transposed*.

4.2.3 The Trace and Partial Trace

String diagrams give us a simple way of sending processes to numbers.

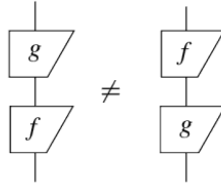
Definition 4.34 For a process f where the input type is the same as the output type, the *trace* is:



and for a process g with one of its inputs having the same type as one of its outputs, the *partial trace* is:

$$\text{tr}_A \left(\begin{array}{c} |A \quad |C \\ \hline \text{g} \\ \hline |A \quad |B \end{array} \right) := \begin{array}{c} \text{C} \\ \hline \text{g} \\ \hline |B \end{array}$$

A funny property of the trace is that while in general:



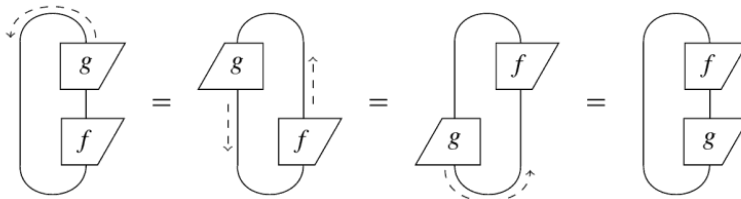
the trace of the LHS and the RHS are in fact equal.

Proposition 4.35 (cyclicity of the trace) We have:

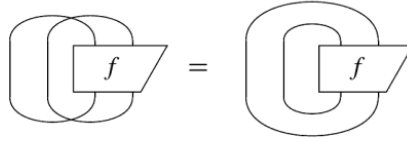
$$\text{tr}(f \circ g) = \text{tr}(g \circ f) \quad \text{i.e.} \quad \begin{array}{c} \text{g} \\ \hline \text{f} \end{array} = \begin{array}{c} \text{f} \\ \hline \text{g} \end{array}$$

Proof This follows from the fact that the two diagrams are equal; i.e. the LHS can be deformed into the RHS without changing how the boxes are wired together. \square

Alternatively, we can give a more step-wise proof of Proposition 4.35 using Proposition 4.30. While not strictly necessary, this other proof is nice because it takes the word ‘cyclicity’ quite literally, yielding a ferris wheel of boxes:



Remark 4.36 In the previous section we decided to distinguish the transpose (which involves nested cups/caps) from the algebraic transpose (which involves criss-crossed cups/caps). Since equation (4.23) also involves cups and caps, we might ask whether for composite systems we need to make a similar distinction between ‘nested’ and ‘criss-crossed’ trace. Luckily, we don’t:



Exercise 4.37 Show that there is only one trace; i.e. any cup/cap pair satisfying the yanking equations (4.11) defines the same trace via (4.23).

4.3 Reflecting Diagrams

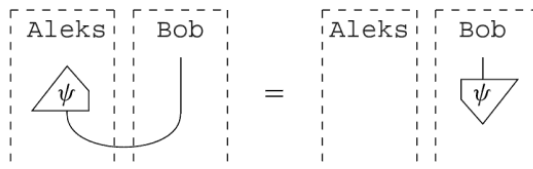
Having string diagrams at hand is already a substantial step towards the quantum world, since it guarantees the existence of non-separable states. They also give rise to some mathematical concepts that play a central role in quantum theory, such as transposition and trace.

We will now identify one more diagrammatic feature that makes string diagrams a lot more articulate, namely *vertical reflection* of diagrams. Vertical reflection allows us to define things like *adjoints*, *conjugates*, *inner products*, *unitarities*, and *positivity*, all of which are major players in any presentation of quantum theory.

Moreover, vertical reflection isn't just some extra degree of freedom that string diagrams happen to have, but has a clear operational meaning in terms of testability of states. This operational meaning will lead to a number of conditions that play an important role in quantum theory. These conditions are in fact quite standard in the literature, but they are usually just stated formally, without any conceptual justification.

4.3.1 Adjoints

In the previous section we explained how the transpose acts graphically by rotating boxes 180°, and that operationally it captures the perfect correlation between Aleks' effects on one side of a cup state and Bob's states on the other:



But what about the relationship between Aleks' effects and Aleks' states?

Recall from Section 3.4.1 that effects can be interpreted as testing a state for some property. Typically, we want to test whether a system is in a particular state. Therefore, we should have a means of relating a state ψ to the effect that tests a system for ψ . String diagrams do not yet tell us how to do this, so we extend our language by representing the effect testing for a state ψ as its vertical reflection:

$$\begin{array}{c} | \\ \hline \nabla \\ \psi \end{array} \mapsto \begin{array}{c} \nabla \\ \psi \\ \hline | \end{array} \tag{4.24}$$

Instead of ‘the effect testing for a state ψ ’ we’ll simply say ψ ’s *adjoint*.

This reflection operation extends very naturally to all processes. If f transforms state ψ into a state ϕ :

$$\begin{array}{c} | \\ \hline \square \\ f \\ \hline \nabla \\ \psi \end{array} = \begin{array}{c} | \\ \hline \nabla \\ \phi \end{array} \tag{4.25}$$

then f ’s *adjoint* is the process that transforms ψ ’s adjoint into ϕ ’s adjoint:

$$\begin{array}{c} \nabla \\ \psi \\ \hline \square \\ f \\ \hline | \end{array} = \begin{array}{c} \nabla \\ \phi \\ \hline | \end{array} \tag{4.26}$$

Note that, as in the case of states, we have depicted the adjoint of an arbitrary process as its vertical reflection:

$$\begin{array}{c} | \\ \hline \square \\ f \\ \hline | \\ A \end{array} \mapsto \begin{array}{c} | \\ \hline \square \\ f \\ \hline | \\ B \end{array}$$

As a result, equation (4.26) is just equation (4.25) upside-down. We use \dagger to denote the operation sending f to its adjoint f^\dagger . As a special case, if we take the adjoint of the effect from (4.24), we get back to ψ :

$$\begin{array}{c} \nabla \\ \psi \\ \hline | \end{array} \mapsto \begin{array}{c} | \\ \hline \nabla \\ \psi \end{array}$$

Hence, like the transpose, adjoints bijectively relate states and effects, and more generally, they bijectively relate processes from a type A to a type B to processes from B to A . Also like the transpose, this is an involution, which is plainly suggested by the diagrammatic notion:

$$\begin{array}{c} | \\ \hline \square \\ f \\ \hline | \end{array} \mapsto \begin{array}{c} | \\ \hline \square \\ f \\ \hline | \end{array} \mapsto \begin{array}{c} | \\ \hline \square \\ f \\ \hline | \end{array} \tag{4.27}$$